Multiple Conclusion Linear Logic: Cut Elimination and more

Harley Eades III[a,*], Valeria de Paiva[b]

[a] *Computer and Information Sciences, Allgood Hall, Augusta University, 120 15th Street, Augusta, GA 30912*
[b] *AI Sunnyvale Lab, Nuance Communications, 1198 E. Arques Ave, Sunnyvale, CA 94085*

Abstract

Full Intuitionistic Linear Logic (FILL) was first introduced by Hyland and de Paiva, and went against current beliefs that it was not possible to incorporate all of the linear connectives, e.g. tensor, par, and implication, into an intuitionistic linear logic. Bierman showed that their formalization of FILL did not enjoy cut-elimination as such, but Bellin proposed a small change to the definition of FILL regaining cut-elimination and using proofnets. In this note we adopt Bellin's proposed change and give a direct proof of cut-elimination for the sequent calculus. Then we show that a categorical model of FILL in the basic dialectica category is also a LinearNonLinear (LNL) model of Benton and a full tensor model of Melliès' and Tabareau's tensorial logic. We give a double-negation translation of linear logic into FILL that explicitly uses par in addition to tensor. Lastly, we introduce a new library to be used in the proof assistant Agda for proving properties of dialectica categories.

Keywords: full intuitionistic linear logic, classical linear logic, dialectica category, cut-elimination, tensorial logic, par, formal proof, proof assistants

1. Introduction

A commonly held belief during the early history of linear logic was that the linear-connective par could not be incorporated into an intuitionistic lin-

*Corresponding author

Email addresses: `heades@augusta.edu` (Harley Eades III), `valeria.depaiva@gmail.com` (Valeria de Paiva)

ear logic. This belief was challenged when de Paiva gave a categorical understanding of Gödel's Dialectica interpretation in terms of dialectica categories [1, 2].

Dialectica categories were originally conceived as models of intuitionistic logic, but they are actually models of intuitionistic linear logic, containing the linear connectives: tensor, implication, the additives, and the exponentials. Further work improved de Paiva's models to capture both intuitionistic and classical linear logic. Armed with this semantic insight de Paiva gave the first formalization of Full Intuitionistic Linear Logic (FILL) [2]. The logical system FILL is a sequent calculus with multiple conclusions in addition to multiple hypotheses. Logics of this type go back to Gentzen's work on the sequent calculus for classical logic LK and for intuitionistic logic LJ, and Maehara's work on LJ' [3, 4]. The sequents in these types of logics usually have the form $\Gamma \vdash \Delta$ where Γ and Δ are multisets of formulas. Sequents such as these are read as "the conjunction of the formulas in Γ imply the disjunction of the formulas in Δ". For a brief, but more complete history of logics with multiple conclusions see the introduction to [5].

Gentzen showed that to obtain intuitionistic logic one could start with the classical logic LK and place a cardinality restriction on the right-hand side of sequents, saying that only one or no formula was allowed in every inference rule. However, this is not the only means of enforcing intuitionism. Maehara showed that in the propositional case one could simply place the cardinality restriction on the premise of the implication right rule, and leave all of the other rules of LK unrestricted. This restriction is sometimes called the Dragalin restriction, as it appeared in his AMS textbook [6]. The classical implication right rule has the form:

$$\frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \multimap B, \Delta}\ \textsc{impR}$$

By placing the Dragalin restriction on the previous rule we obtain:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ \textsc{impR}$$

The first formalization of FILL used the Dragalin restriction, see [2] p. 58, but Schellinx showed that this restriction has the unfortunate consequence of breaking cut-elimination [7].

Hyland and de Paiva gave an alternative formalization of FILL with the intention of regaining cut-elimination [8]. This new formalization lifted the Dragalin restriction by decorating sequents with a term assignment. Hypotheses were assigned variables, and the conclusions were assigned terms. Then using these terms one can track the use of hypotheses throughout a derivation. They proposed a new implication right rule:

$$\frac{\Gamma, x : A \vdash t : B, \Delta \qquad x \notin \mathsf{FV}(\Delta)}{\Gamma \vdash \lambda x.t : A \multimap B, \Delta}\ \textsc{impR}$$

Intuitionism is enforced in this rule by requiring that the variable being discharged, x, is not free in terms annotating other conclusions. This formalization did not enjoy cut-elimination either.

Bierman was able to give a counterexample to cut-elimination [9]. As Bierman explains the problem then was with the left rule for the multiplicative disjunction par. The original rule was as follows:

$$\frac{\Gamma, x : A \vdash \Delta \qquad \Gamma', y : B \vdash \Delta'}{\Gamma, \Gamma', z : A \parr B \vdash \mathsf{let}\ z\ \mathsf{be}\ (x \parr -)\ \mathsf{in}\ \Delta \mid \mathsf{let}\ z\ \mathsf{be}\ (- \parr y)\ \mathsf{in}\ \Delta'}\ \textsc{parL}$$

In this rule the pattern variables x and y are bound in each term of Δ and Δ' respectively. Notice that the variable z becomes free in every term in Δ and Δ'. Bierman showed that this rule mixed with the restriction on implication right prevents the usual cut-elimination step that commutes cut with the left rule for par. The main idea behind the counterexample is that in the derivation before commuting the cut it is possible to discharge z using implication right, but after the cut is commuted past the left rule for par, the variable z becomes free in more than one conclusion, and thus, can no longer be discharged.

In the conclusion of Bierman's note he describes Bellin's alternate left rule for par. This new left-rule is as follows:

$$\frac{\Gamma, x : A \vdash \Delta \quad \Gamma', y : B \vdash \Delta'}{\Gamma, \Gamma', z : A \parr B \vdash \text{let-pat}\, z\, (x \parr -)\, \Delta \mid \text{let-pat}\, z\, (- \parr y)\, \Delta'}\ \textsc{Parl}$$

In this rule let-pat $z\,(x \parr -)\,t$ and let-pat $z\,(- \parr y)\,t'$ only let-bind z in t or t' if $x \in FV(t)$ or $y \in FV(t')$. Otherwise the terms are left unaltered. Bellin showed that by adopting this rule cut-elimination can be proven by reduction to the cut-elimination procedure for proof nets for multiplicative linear logic with the mix rule [10]. However, this is an indirect proof that requires the adoption of proof nets.

Contributions. In this paper our main contribution is to give a direct proof of cut-elimination for FILL with Bellin's proposed par-left rule (Section 3). A direct proof accomplishes two goals: the first is to complete the picture of FILL Hyland and de Paiva started, and the second is to view a direct proof of cut-elimination as a means of checking the correctness of the formulation of FILL given here. The latter point is important for future work. Following the proof of cut-elimination we show that the categorical model of FILL called $\mathsf{Dial}_2(\mathsf{Sets})$, the basic dialectica category, is also a linear/non-linear model of Benton (Section 4) and a full tensor model of Melliès' and Tabareau's tensor logic (Section 5). Then we give a double-negation translation of multi-conclusion classical linear logic into FILL (Section 5.1). Due to the complexities of working in $\mathsf{Dial}_2(\mathsf{Sets})$ we have formalized all of the constructions and proofs used in Section 4 and Section 5 – although our formal verification does not include the double-negation translation in Section 5.1 – in the Agda proof assistant[1]. We synthetized a general library for proving properties of dialectica categories and introduce it in Section 6 along with generalizations of dialectica categories using the notion of a lineale.

Related Work. The first formalization of FILL with cut-elimination was due to Braüner and de Paiva [11]. Their formalization can be seen as a linear version of LK with a sophisticated meta-level dependency tracking system. A proof of a FILL sequent in their formalization amounts to a classical derivation, π, invariant in what they call the FILL property:

- The hypothesis discharged by an application of the implication right rule in π is a dependency of the conclusion of the implication being introduced.

They were able to show that their formalization is sound, complete, and enjoys cut-elimination. In favor of the term assignment formalization given here over Braüner and de Paiva's formalization we can say that the dependency tracking system complicates both the definition of the logic and its use. However, one might conjecture that their system is more fundamental and hence more generalizable. It might be possible to prove cut-elimination

[1]The Agda development can be found at `https://github.com/heades/cut-fill-agda`.

of the term assignment formalization of FILL relative to Braüner and de Paiva's dependency tracking system by erasing the terms on conclusions and then tracking which variable is free in which conclusion. However, as we stated above a direct proof is more desirable than a relative one.

The work of de Paiva and Pereira used annotations on the sequents of LK to arrive at full intuitionistic logic (FIL) with multiple conclusion that enjoys cut-elimination [5]. They annotate hypothesis with natural number indices, and conclusions with finite sets of indices. The sets of indices on conclusions correspond to the collection of the hypotheses that the conclusion depends on. Then they have a similar property to that of Braüner and de Paiva's formalization. In fact, the dependency tracking system is very similar to this formalization, but the dependency tracking has been collapsed into the object language instead of being at the meta-level.

Clouston et al. give both a deep inference calculus and a display calculus for FILL that admits cut-elimination [12]. Both of these systems are refinements of a larger one called bi-intuitionistic linear logic (BiLL). This logic contains every logical connective of FILL with the addition of the exclusion (or subtraction) connective. This connective can be defined categorically as the left-adjoint to par. Thus, exclusion is the dual to implication. A positive aspect to this work is that the resulting systems are annotation free, but at a price of complexity. Deep inference and display calculi are harder to understand, and their system requires FILL to be defined as a refinement of a system with additional connectives. We show in this paper that such a refinement is unnecessary. In addition, a term assignment system is closer to traditional logic than deep inference and display calculi, and it is closer, through the lens of the Curry-Howard-Lambek correspondence, to a type theoretic understanding of FILL.

2. Full Intuitionistic Linear Logic (FILL)

In this section we give a brief description of FILL. We first give the syntax of formulas, patterns, terms, and contexts. Following the syntax we define several meta-functions that will be used when defining the inference rules of the logic.

Definition 1. The syntax for FILL is as follows:

(Formulas)	$A, B, C, D, E ::= \top \mid \perp \mid A \multimap B \mid A \otimes B \mid A \parr B$
(Patterns)	$p ::= * \mid - \mid x \mid p_1 \otimes p_2 \mid p_1 \parr p_2$
(Terms)	$t, e ::= x \mid * \mid \circ \mid t_1 \otimes t_2 \mid t_1 \parr t_2 \mid \lambda x.t \mid \mathsf{let}\, t\, \mathsf{be}\, p\, \mathsf{in}\, e \mid t_1\, t_2$
(Left Contexts)	$\Gamma ::= \cdot \mid x : A \mid \Gamma_1, \Gamma_2$
(Right Contexts)	$\Delta ::= \cdot \mid t : A \mid \Delta_1, \Delta_2$

The formulas of FILL are standard, but we denote the unit of tensor as $\top$ and the unit of par as $\perp$. Patterns are used to distinguish between the various let-expressions for tensor, par, and their units. There are three different let-expressions:

Tensor:	Par:	Tensor Unit:
$\mathsf{let}\, t\, \mathsf{be}\, p_1 \otimes p_2\, \mathsf{in}\, e$	$\mathsf{let}\, t\, \mathsf{be}\, p_1 \parr p_2\, \mathsf{in}\, e$	$\mathsf{let}\, t\, \mathsf{be}\, *\, \mathsf{in}\, e$

In addition, each of these will have their own equational rules, see Figure 2. The role each term plays in the overall logic will become clear after we introduce the inference rules.

At this point we introduce some syntax and meta-level functions that will be used in the definition of the inference rules for FILL. Left contexts are multisets of formulas each labeled with a variable, and right contexts are multisets of formulas each labeled with a term. We will often write $\Delta_1 \mid \Delta_2$ as syntactic sugar for Δ_1, Δ_2. The former should be read as "Δ_1 or Δ_2." We denote the usual capture-avoiding substitution by $[t/x]t'$, and its straightforward extension to right contexts as $[t/x]\Delta$. Similarly, we find it convenient to be able to do this style of extension for the let-binding as well.

Definition 2. We extend let-binding terms to right contexts as follows:

$$\begin{aligned}
&\mathsf{let}\, t\, \mathsf{be}\, p\, \mathsf{in} \cdot = \cdot \\
&\mathsf{let}\, t\, \mathsf{be}\, p\, \mathsf{in}\, (t' : A) = (\mathsf{let}\, t\, \mathsf{be}\, p\, \mathsf{in}\, t') : A \\
&\mathsf{let}\, t\, \mathsf{be}\, p\, \mathsf{in}\, (\Delta_1 \mid \Delta_2) = (\mathsf{let}\, t\, \mathsf{be}\, p\, \mathsf{in}\, \Delta_1) \mid (\mathsf{let}\, t\, \mathsf{be}\, p\, \mathsf{in}\, \Delta_2)
\end{aligned}$$

Lastly, we denote the usual function that computes the set of free variables in a term by $\mathsf{FV}(t)$, and its straightforward extension to right contexts as $\mathsf{FV}(\Delta)$.

The inference rules for FILL are defined in Figure 1. The par left ParL rule depends on the function let-pat $z\ p\ \Delta$ which we define next.

Definition 3. The function let-pat $z\ p\ t$ is defined as follows:

$$\frac{}{x : A \vdash x : A}\ \textsc{Ax} \qquad \frac{\Gamma \vdash t : A \mid \Delta \quad \Gamma', y : A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta \mid [t/y]\Delta'}\ \textsc{Cut} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, x : \top \vdash \mathsf{let}\, x\, \mathsf{be}\, *\, \mathsf{in}\, \Delta}\ \textsc{Tl}$$

$$\frac{}{\cdot \vdash * : \top}\ \textsc{Tr} \qquad \frac{\Gamma, x : A, y : B \vdash \Delta}{\Gamma, z : A \otimes B \vdash \mathsf{let}\, z\, \mathsf{be}\, x \otimes y\, \mathsf{in}\, \Delta}\ \textsc{Tenl}$$

$$\frac{\Gamma \vdash e : A \mid \Delta \quad \Gamma' \vdash f : B \mid \Delta'}{\Gamma, \Gamma' \vdash e \otimes f : A \otimes B \mid \Delta \mid \Delta'}\ \textsc{Tenr} \qquad \frac{}{x : \perp \vdash \cdot}\ \textsc{Pl} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \circ : \perp \mid \Delta}\ \textsc{Pr}$$

$$\frac{\Gamma, x : A \vdash \Delta \quad \Gamma', y : B \vdash \Delta'}{\Gamma, \Gamma', z : A \parr B \vdash \text{let-pat}\, z\, (x \parr -)\, \Delta \mid \text{let-pat}\, z\, (- \parr y)\, \Delta'}\ \textsc{Parl}$$

$$\frac{\Gamma \vdash \Delta \mid e : A \mid f : B \mid \Delta'}{\Gamma \vdash \Delta \mid e \parr f : A \parr B \mid \Delta'}\ \textsc{Parr} \qquad \frac{\Gamma \vdash e : A \mid \Delta \quad \Gamma', x : B \vdash \Delta'}{\Gamma, y : A \multimap B, \Gamma' \vdash \Delta \mid [y\, e/x]\Delta'}\ \textsc{Impl}$$

$$\frac{\Gamma, x : A \vdash e : B \mid \Delta \quad x \notin \mathsf{FV}(\Delta)}{\Gamma \vdash \lambda x.e : A \multimap B \mid \Delta}\ \textsc{Impr} \qquad \frac{\Gamma, x : A, y : B \vdash \Delta}{\Gamma, y : B, x : A \vdash \Delta}\ \textsc{Exl}$$

$$\frac{\Gamma \vdash \Delta_1 \mid t_1 : A \mid t_2 : B \mid \Delta_2}{\Gamma \vdash \Delta_1 \mid t_2 : B \mid t_1 : A \mid \Delta_2}\ \textsc{Exr}$$

Figure 1: Inference rules for FILL

$$\begin{array}{ccc} \text{let-pat}\, z\, (x \parr -)\, t = t & \text{let-pat}\, z\, (- \parr y)\, t = t & \text{let-pat}\, z\, p\, t = \mathsf{let}\, z\, \mathsf{be}\, p\, \mathsf{in}\, t \\ \text{where } x \notin \mathsf{FV}(t) & \text{where } y \notin \mathsf{FV}(t) & \end{array}$$

It is straightforward to extend the previous definition to right-contexts, and we denote this extension by let-pat $z\, p\, \Delta$.

The motivation behind this function is that it only binds the pattern variables in $x \parr -$ and $- \parr y$ if and only if those pattern variables are free in the body of the let. This overcomes the counterexample given by Bierman in [9].

The terms of FILL are equipped with an equivalence relation defined in Figure 2. There are a number of α, β, and η like rules as well as several rules we call naturality rules. These rules are similar to the rules presented in [8].

3. Cut-elimination

The system FILL can be viewed from two different angles: i. as an intuitionistic linear logic with par, or ii. as a restricted form of classical linear logic. Thus, to prove cut-elimination of FILL one only needs to start with the cut-elimination procedure for intuitionistic linear logic, and then dualize all of the steps in the procedure for tensor and its unit to obtain the steps

$$\frac{y \notin \mathsf{FV}(t)}{t = [y/x]t}\ \textsc{Alpha} \qquad \frac{x \notin \mathsf{FV}(f)}{(\lambda x.f\ x) = f}\ \textsc{EtaFun} \qquad \frac{}{(\lambda x.e)\ e' = [e'/x]e}\ \textsc{BetaFun}$$

$$\frac{}{\mathsf{let}\ *\ \mathsf{be}\ *\ \mathsf{in}\ e = e}\ \textsc{Eta1I} \qquad \frac{}{\mathsf{let}\ u\ \mathsf{be}\ *\ \mathsf{in}\ [*/z]f = [u/z]f}\ \textsc{BetaI}$$

$$\frac{}{[\mathsf{let}\ u\ \mathsf{be}\ *\ \mathsf{in}\ e/y]f = \mathsf{let}\ u\ \mathsf{be}\ *\ \mathsf{in}\ [e/y]f}\ \textsc{NatI}$$

$$\frac{}{\mathsf{let}\ e \otimes t\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ u = [e/x, t/y]u}\ \textsc{Beta1Ten}$$

$$\frac{}{\mathsf{let}\ u\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ [x \otimes y/z]f = [u/z]f}\ \textsc{Beta2Ten}$$

$$\frac{}{[\mathsf{let}\ u\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ g/w]f = \mathsf{let}\ u\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ [g/w]f}\ \textsc{NatTen} \qquad \frac{}{u = \circ}\ \textsc{EtaParU}$$

$$\frac{}{(\mathsf{let}\ u\ \mathsf{be}\ x \parr -\ \mathsf{in}\ x) \parr (\mathsf{let}\ u\ \mathsf{be}\ - \parr y\ \mathsf{in}\ y) = u}\ \textsc{EtaPar}$$

$$\frac{}{\mathsf{let}\ u \parr t\ \mathsf{be}\ x \parr -\ \mathsf{in}\ e = [u/x]e}\ \textsc{Beta1Par} \qquad \frac{}{\mathsf{let}\ u \parr t\ \mathsf{be}\ - \parr y\ \mathsf{in}\ e = [t/y]e}\ \textsc{Beta2Par}$$

$$\frac{}{\mathsf{let}\ t\ \mathsf{be}\ x \parr -\ \mathsf{in}\ [u/x]f = [\mathsf{let}\ t\ \mathsf{be}\ x \parr -\ \mathsf{in}\ u/x]f}\ \textsc{Nat1Par}$$

$$\frac{}{\mathsf{let}\ t\ \mathsf{be}\ - \parr y\ \mathsf{in}\ [v/y]f = [\mathsf{let}\ t\ \mathsf{be}\ - \parr y\ \mathsf{in}\ v/y]f}\ \textsc{Nat2Par}$$

Figure 2: Equivalence on terms

for par and its unit. Similarly, one could just as easily start with the cut-elimination procedure for classical linear logic, and then apply the restriction on the implication right rule producing a cut-elimination procedure for FILL.

The major difference between proving cut-elimination of FILL from classical or intuitionistic linear logic is that we must prove an invariant across each step in the procedure. The invariant is that if a derivation π is transformed into a derivation π', then the terms in the conclusion of the final rule applied in π must be be transformable, when the equivalences defined in Figure 2 are taken as left-to-right rewrite rules, into the terms in the conclusion of the final rule applied in π'.

We finally arrive at cut-elimination.

Theorem 1. *If* $\Gamma \vdash t_1 : A_1, \ldots, t_i : A_i$ *steps to* $\Gamma \vdash t'_1 : A_1, \ldots, t'_i : A_i$ *using* *the cut-elimination procedure, then $t_j = t'_j$ for $1 \leq j \leq i$.*

Proof. The cut-elimination procedure given here is the standard cut-elimination procedure for classical linear logic except that the cases involving the implication right rule have the FILL restriction. The structure of our procedure follows the structure of the procedure found in [13]. Throughout this proof we treat the equivalences defined in Figure 2 as left-to-right rewrite rules. For the entire proof see the companion report [14].

Corollary 2 (Cut-Elimination). *Cut-elimination holds for FILL.*

4. Full LNL Models

One of the difficult questions considering the categorical models of linear logic was how to model Girard's exponential, !, which is read "of course". The ! modality can be used to translate intuitionistic logic into intuitionistic linear logic, and so the correct categorical interpretation of ! should involve a relationship between a cartesian closed category, and the model of intuitionistic linear logic.

The work of de Paiva gave some of the very first categorical models for both classical and intuitionistic linear logic in the thesis [2]. She showed that a particular dialectica category called $\mathsf{Dial}_2(\mathsf{Sets})$ is a model of FILL where ! is interpreted as a comonad which produces natural comonoids, see page 76 of [2].

Definition 4. The category $\mathsf{Dial}_2(\mathsf{Sets})$ consists of

- objects that are triples, $A = (U, X, \alpha)$, where U and X are sets, and $\alpha \subseteq U \times X$ is a relation, and
- maps that are pairs $(f, F) : (U, X, \alpha) \longrightarrow (V, Y, \beta)$ where $f : U \longrightarrow V$ and $F : Y \longrightarrow X$ such that
  - For any $u \in U$ and $y \in Y$, $\alpha(u, F(y))$ implies $\beta(f(u), y)$.

Suppose $A = (U, X, \alpha)$, $B = (V, Y, \beta)$, and $C = (W, Z, \gamma)$. Then identities are given by $(\mathsf{id}_U, \mathsf{id}_X) : A \longrightarrow A$. The composition of the maps $(f, F) : A \longrightarrow B$ and $(g, G) : B \longrightarrow C$ is defined as $(f; g, G; F) : A \longrightarrow C$.

In her thesis de Paiva defines a particular class of dialectica categories called GC over a base category C, see page 41 of [2]. The category $\mathsf{Dial}_2(\mathsf{Sets})$ defined above can be seen as an instantiation of GC by setting C to be the category Sets of sets and functions between them. This model is a non-trivial model (all four units of the multiplicative and additive conjunction and disjunction are different objects in the category), and does not model classical logic; see [2] page 58.

Seely gave a different, syntactic categorical model that confirmed that the of-course exponential should be modeled by a comonad [15]. However, Seely's model turned out to be unsound, as pointed out by Bierman [16]. This was noticed when Benton, Bierman, de Paiva, and Hyland, worked out another categorical model called linear categories (Definition 5) that are sound, and also model ! using a monoidal comonad [16].

Definition 5. A **linear category**, $\mathcal{L}$, consists of:

- A symmetric monoidal closed category $\mathcal{L}$,
- A symmetric monoidal comonad $(!, \epsilon, \delta, m_{A,B}, m_I)$ such that
  - For every free !-coalgebra $(!A, \delta_A)$ there are two distinguished monoidal natural transformations $e_A : !A \longrightarrow I$ and $d_A : !A \longrightarrow !A \otimes !A$ which form a commutative comonoid and are coalgebra morphisms.
  - If $f : (!A, \delta_A) \longrightarrow (!B, \delta_B)$ is a coalgebra morphism between free coalgebras, then it is also a comonoid morphism.

This definition is the one given by Bierman in his thesis, see [16] for full definitions.

Intuitionistic logic can be interpreted in a linear category as a full subcategory of the category of !-coalgebras for the comonad, see proposition 17 of [16].

Benton gave a more balanced view of linear categories called LNL models.

Definition 6. A **linear/non-linear model (LNL model)** consists of

- a cartesian closed category $(\mathcal{C}, 1, \times, \Rightarrow)$,
- a SMCC $(\mathcal{L}, I, \otimes, \multimap)$, and

- a pair of symmetric monoidal functors $(G, n) : \mathcal{L} \longrightarrow \mathcal{C}$ and $(F, m) : \mathcal{C} \longrightarrow \mathcal{L}$ between them that form a symmetric monoidal adjunction with $F \dashv G$.

See Benton, [17], for the definitions of symmetric monoidal functors and adjunctions.

A non-trivial consequence of the definition of a LNL model is that the ! modality can indeed be interpreted as a monoidal comonad. Suppose $(\mathcal{L}, \mathcal{C}, F, G)$ is a LNL model. Then the comonad is given by $(!, \epsilon : ! \longrightarrow \mathsf{Id}, \delta : ! \longrightarrow !!)$ where $! = FG$, ϵ is the counit of the adjunction and δ is the natural transformation $\delta_A = F(\eta_{G(A)})$, see page 15 of [17]. We recall the following result from Benton [17]:

Theorem 3 (LNL Models and Linear Categories).

i. (Section 2.2.1 of [17]) Every LNL model is a linear category.

ii. (Section 2.2.2 of [17]) Every linear category is a LNL model.

Proof. The proof of part i. is a matter of checking that each part of the definition of a linear category can be constructed using the definition of a LNL model. See lemmata 3-7 of [17].

As for the proof of part ii. Given a linear category we have a SMCC and so the difficulty of proving this result is constructing the CCC and the adjunction between both parts of the model. Suppose $\mathcal{L}$ is a linear category. Benton constructs the CCC out of the full subcategory of Eilenberg-Moore category $\mathcal{L}^!$ whose objects are exponentiable coalgebras denoted $\mathsf{Exp}(\mathcal{L}^!)$. He shows that this subcategory is cartesian closed, and contains the (co)Kleisli category, $\mathcal{L}_!$, see Lemma 11 on page 23 of [17]. The required adjunction $F : \mathsf{Exp}(\mathcal{L}^!) \longrightarrow L : G$ can be defined using the adjunct functors $F(A, h_A) = A$ and $G(A) = (!A, \delta_A)$, see lemmata 13 - 16 of [17].

While this theorem is totally satisfactory for our goals in this paper, one should perhaps remark that it does not answer the harder question of which kind of morphisms should one consider between models of Linear Logic. A fuller discussion can be found in [18].

Next we show that the category $\mathsf{Dial_2(Sets)}$ is a full version of a linear category. First, we extend the definitions of linear categories and LNL models to be equipped with the necessary categorical structure to model par and its unit.

Definition 7. A **full linear category**, $\mathcal{L}$, consists of a linear category $(\mathcal{L}, \top, \otimes, \multimap, !A, e_A, d_A)$, a symmetric monoidal structure on L, $(\perp, \parr)$, and distribution natural transformations $\mathsf{dist}_1 : A \otimes (B \parr C) \longrightarrow (A \otimes B) \parr C$ and $\mathsf{dist}_2 : (A \parr B) \otimes C \longrightarrow A \parr (B \otimes C)$. The distributors must satisfy several coherence conditions which can be found in [19].

Definition 8. A **full linear/non-linear model (full LNL model)** consists of a LNL model $(\mathcal{L}, \mathcal{C}, F, G)$, and a symmetric monoidal structure on L, $(\perp, \parr)$, as above.

First we show that $\mathsf{Dial}_2(\mathsf{Sets})$ is a full linear category, and then using the proof by Benton that linear categories are LNL models we obtain that $\mathsf{Dial}_2(\mathsf{Sets})$ is a full LNL model. In order for this to work we need to know that $\mathsf{Dial}_2(\mathsf{Sets})$ has a symmetric *monoidal* comonad $(!, \epsilon, \delta, m_{A,B}, m_I)$. At the time of de Paiva's thesis it was not known that the comonad modeling the of-course modality needed to be *monoidal*.

The comonad $(!, \epsilon, \delta, m_{A,B}, m_I)$ can be defined by setting $!(U, X, \alpha) = (U, U \longrightarrow X^*, \alpha*)$ where X^* the commutative monoid consisting of finite sequences of elements of X, and $\alpha^* \subseteq U \times X^*$ is the extension of α to finite sequences, that is, $\alpha^*(u, f)$ iff $\alpha(u, x_1) \wedge \cdots \wedge \alpha(u, x_n)$, where $f(u) = x_1, \ldots, x_n$. The definitions of ϵ and δ can be found in the formal development.

We can show that the monoidal maps $m_{A,B} :!A \otimes !B \longrightarrow !(A \otimes B)$ and $m_I : I \longrightarrow !I$ exist in the general setting of dialectica categories, and thus, these maps exist in $\mathsf{Dial}_2(\mathsf{Sets})$. Intuitively, given two objects $A = (X, U, \alpha)$ and $B = (V, Y, \beta)$ of $\mathsf{Dial}_2(\mathsf{Sets})$ the map $m_{A,B}$ is defined as the pair $(\mathsf{id}_{U \times V}, F)$, where $F = (F_1, F_2)$, $F_1 : (U \times V) \Rightarrow (V \Rightarrow X)^* \longrightarrow V \Rightarrow (U \Rightarrow X^*)$ and $F_2 : (U \times V) \Rightarrow (U \Rightarrow Y)^* \longrightarrow U \Rightarrow (V \Rightarrow Y^*)$. The maps F_1 and F_2 build the sequence of all the results of applying each function in the input sequence to the input coordinate.

We can now show that the following holds.

Lemma 4. *The category* $\mathsf{Dial}_2(\mathsf{Sets})$ *is a full linear category.*

Proof. We only give a sketch of the proof here, but for the full details see that companion report [14][2]. First, we must show that $\mathsf{Dial}_2(\mathsf{Sets})$ is a

[2]This proof was formalized in the Agda proof assistant see the file `https://github.com/heades/cut-fill-agda/blob/master/FullLinCat.agda`

linear category. The majority of the linear structure of $\mathsf{Dial}_2(\mathsf{Sets})$ is in de Paiva's thesis [2]. We had to extend her definitions to show that the comonad $(!A, \delta, \epsilon)$ is monoidal. However, this extension is straightforward.

After showing that $\mathsf{Dial}_2(\mathsf{Sets})$ is a linear category one must show that $\mathsf{Dial}_2(\mathsf{Sets})$ is a model of par and its unit. This easily follows from de Paiva's thesis. The bifunctor which models par is given by de Paiva in Definition 10 on page 47 of [2].

Finally, $\mathsf{Dial}_2(\mathsf{Sets})$ must be distributive. The natural transformations $dist_1$ and $dist_2$ can be defined in terms of the maps $k : (A \otimes A') \otimes (B \parr C) \longrightarrow (A \otimes B) \parr (A' \otimes C)$ and $k' : (A \parr B) \otimes (C \otimes C') \longrightarrow (A \otimes C) \parr (B \otimes C')$ given on page 52 of [2]. Set $A' = \top$ in k and $C = \top$ in k' to obtain $dist_1$ and $dist_2$ respectively. They can also be shown to satisfy the coherence conditions given in [19].

Corollary 5. *The category* $\mathsf{Dial}_2(\mathsf{Sets})$ *is a full LNL model.*

Proof. This follows directly from the previous lemma and Theorem 3 which shows that linear categories are LNL models.

The point of these calculations is to show that the several different axiomatizations available for models for linear logic are consistent and that a model proved sound and complete according to Seely's definition (using the Seely isomorphisms $!(A \times B) \cong !A \otimes !B$ and $!1 \cong \top$ but adding to it monoidicicty of the comonad) is indeed sound and complete as a LNL model too.

5. Tensorial Logic

Melliès and Tabareau introduced tensorial logic [20] as a means of generalizing linear logic to a theory of tensor and a non-involutive negation called tensorial negation. That is, instead of an isomorphism $A = \neg\neg A$ we have only a natural transformation $A \longrightarrow \neg\neg A$, just as one does in FILL. Tensorial logic makes the claim that tensor and tensorial negation are more fundamental than tensor and negation defined via implication into false, as in FILL. This is at odds with FILL where implication is considered fundamental.

In this section we show that multiplicative tensorial logic can be modeled by $\mathsf{Dial}_2(\mathsf{Sets})$ (Lemma 7) by showing that tensorial negation arises as a simple property of the implication in any symmetric monoidal closed category (Lemma 6). While this is expected (after all negation being defined in terms of implication into absurdity is one of the staples of intuitionism) we think it bolsters our claim that linear implication is a fundamental connective that should not be redefined in terms of the multiplicative disjunction par. In any case, any model of FILL can be seen as a model of multiplicative tensorial logic.

A categorical model of tensorial logic is a symmetric monoidal category with a tensorial negation.

Definition 9. A **tensorial negation** on a symmetric monoidal category $(\mathcal{C}, \otimes, I)$ is defined as a functor $\neg : \mathcal{C} \longrightarrow \mathcal{C}^{\mathsf{op}}$ together with a family of bijections $\phi_{A,B,C} : \mathsf{Hom}_{\mathcal{C}}(A \otimes B, \neg C) \cong \mathsf{Hom}_{\mathcal{C}}(A, \neg(B \otimes C))$ natural in A, B, and C. Furthermore, the following diagram must commute:

$$\begin{tikzcd}
\mathsf{Hom}(A \otimes (B \otimes C), \neg D) \arrow[rr, "\mathsf{Hom}(\alpha_{A,B,C}, \mathsf{id}_{\neg D})"] \arrow[dd, "\phi_{A,B\otimes C,D}"'] & & \mathsf{Hom}((A \otimes B) \otimes C, \neg D) \arrow[d, "\phi_{A\otimes B,C,D}"] \\
& & \mathsf{Hom}(A \otimes B, \neg(C \otimes D)) \arrow[d, "\phi_{A,B,C\otimes D}"] \\
\mathsf{Hom}(A, \neg((B \otimes C) \otimes D)) \arrow[rr, "\mathsf{Hom}(\mathsf{id}_A, \neg\alpha_{B,C,D})"'] & & \mathsf{Hom}(A, \neg(B \otimes (C \otimes D)))
\end{tikzcd}$$

The most basic form of tensorial logic is called multiplicative tensorial logic and only consists of a tensor and a tensorial negation. The model of multiplicative tensorial logic is called a dialogue category.

Definition 10. A **dialogue category** is a symmetric monoidal category equipped with a tensorial negation.

We show that tensorial negation arises as a simple property of implication, as is traditional.

Lemma 6. *In any monoidal closed category, $\mathcal{C}$, there is a natural bijection $\phi_{A,B,C,D} : \mathsf{Hom}_{\mathcal{C}}(A \otimes B, C \multimap D) \cong \mathsf{Hom}_{\mathcal{C}}(A, (B \otimes C) \multimap D)$. Furthermore, the following diagram commutes:*

$$\begin{tikzcd}
\mathsf{Hom}(A \otimes (B \otimes C), D \multimap E) \arrow[rr, "\mathsf{Hom}(\alpha_{A,B,C}, \mathsf{id}_{D \multimap E})"] \arrow[dd, "\phi_{A,B\otimes C,D,E}"'] & & \mathsf{Hom}((A \otimes B) \otimes C, D \multimap E) \arrow[d, "\phi_{A\otimes B,C,D,E}"] \\
& & \mathsf{Hom}(A \otimes B, (C \otimes D) \multimap E) \arrow[d, "\phi_{A,B,C\otimes D,E}"] \\
\mathsf{Hom}(A, ((B \otimes C) \otimes D) \multimap E) \arrow[rr, "\mathsf{Hom}(\mathsf{id}_A, \alpha_{B,C,D} \multimap E)"'] & & \mathsf{Hom}(A, (B \otimes (C \otimes D)) \multimap E)
\end{tikzcd}$$

Proof. Suppose $\mathcal{C}$ is a monoidal closed category. Then we can define $\phi(f : A \otimes B \longrightarrow C \multimap D) = \mathsf{cur}(\alpha^{-1}; \mathsf{cur}^{-1}(f))$ and $\phi^{-1}(g : A \longrightarrow (B \otimes C) \multimap D) = \mathsf{cur}(\alpha; \mathsf{cur}^{-1}(g))$. Clearly, these are mutual inverses, and hence, ϕ is a bijection. Naturality of ϕ easily follows. Lastly, the diagram given above also commutes. For the complete proof see the companion report [14].

Any model of FILL contains the unit of par, $\perp$, and thus, can be used to define the negation function $\neg A := A \multimap \perp$. Now replacing D and E in the previous result with $\perp$ yields the definition of tensorial negation.

Lemma 7. $\mathsf{Dial}_2(\mathsf{Sets})$ *is a model of multiplicative tensorial logic.*

Proof. We have already shown $\mathsf{Dial}_2(\mathsf{Sets})$ to be a model of FILL, and thus, it has a symmetric monoidl closed structure as well as the negation functor, and thus, by Lemma 6 it has a tensorial negation[3].

Extending a model of multiplicative tensorial logic with an exponential resource modality yields a model of full tensorial logic.

Definition 11. A **resource modality** on a symmetric monoidal category $(\mathcal{C}, \otimes, I)$ is an adjunction with a symmetric monoidal category $(\mathcal{M}, \otimes', I')$:

$$\begin{tikzcd} \mathcal{M} \arrow[r, bend left, "F"] & \mathcal{C} \arrow[l, bend left, "G"] \end{tikzcd}$$

A resource modality is called an **exponential resource modality** if $\mathcal{M}$ is cartesian where $\otimes'$ is the product and I' is the terminal object.

A model of full tensorial logic is defined to be a model of multiplicative tensorial logic with an exponential resource modality. We now know that $\mathsf{Dial}_2(\mathsf{Sets})$ is a model of multiplicative tensorial logic. By constructing the co-Kleisli category which consists of the !-coalgebras as objects, and happens to be cartesian, we can show that $\mathsf{Dial}_2(\mathsf{Sets})$ is a model of full tensorial logic. The adjunction with the co-Kleisli category naturally arises from the proof that $\mathsf{Dial}_2(\mathsf{Sets})$ is a full LNL model (Corollary 5).

[3] We give a full proof in the formalization see the file `https://github.com/heades/cut-fill-agda/blob/master/Tensorial.agda`.

Lemma 8. *The category* $\mathsf{Dial_2(Sets)}$ *is a model of full tensorial logic.*

PROOF. It suffices to show that there is an adjunction between $\mathsf{Dial_2(Sets)}$ and a cartesian category. Define the category $\mathsf{Dial_2(Sets)_!}$ as follows:

- Take as objects $(U, (U \Rightarrow X^*), \alpha_!)$ where U and X are sets, and $\alpha \subseteq U \times (U \Rightarrow X^*)$.
- Take as morphisms $(f, F) : (U, (U \Rightarrow X^*), \alpha_!) \longrightarrow (V, (V \Rightarrow Y^*), \beta_!)$ where $f : U \longrightarrow V$ and $F : (V \Rightarrow Y^*) \longrightarrow (U \Rightarrow X^*)$ subject to the same condition on morphisms as $\mathsf{Dial_2(Sets)}$. Composition and identities are defined similarly to $\mathsf{Dial_2(Sets)}$.

Next we must show that $\mathsf{Dial_2(Sets)_!}$ is cartesian. Notice that $\mathsf{Dial_2(Sets)_!}$ is a subcategory of $\mathsf{Dial_2(Sets)}$, and there is a functor $J : \mathsf{Dial_2(Sets)} \longrightarrow \mathsf{Dial_2(Sets)_!}$ which is defined equivalently to the endofunctor ! from the proof of Lemma 4. In fact, $\mathsf{Dial_2(Sets)_!}$ is the co-Kleisli category with objects free !-coalgebras and is cartesian closed [1]. However, we only need the fact that it is cartesian.

To show that $\mathsf{Dial_2(Sets)_!}$ is cartesian it suffices to show that J preserves the cartesian structure of $\mathsf{Dial_2(Sets)}$ – the proof that $\mathsf{Dial_2(Sets)}$ is cartesian can be found on page 48 of [2]. This follows by straightforward reasoning. For the complete proof see the companion report [14].

5.1. Double Negation Translation

In this section we show that we can use intuitionistic negation – which we showed was tensorial in the previous section – to define a negative translation of multi-conclusion linear logic (Figure 3) into FILL where implication plays a central role. Melliès and Tabareau give a negative translation of the multiplicative fragment of linear logic into tensorial logic [21] using tensor as the main connective. For example, they define $(A \otimes B)^N = \neg(\neg(A)^N \otimes \neg(B)^N)$, and thus, they simulate par using tensor and negation. This definition would cause some syntactic issues with FILL, because the left-rule to par requires the let-pattern term defined in Definition 3, thus, encoding par in terms of tensor would require the let-pattern term to be used in the left-rule for tensor. While simulating par, using tensor and negation, can be seen as useful, in applications where only the tensor product can be actually calculated, in other applications we do have an extra bifunctor like par. This is true in the

$$\frac{}{A \vdash A}\ \textsc{LL\_Ax} \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ \textsc{LL\_Cut} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, \top \vdash \Delta}\ \textsc{LL\_Tl}$$

$$\frac{}{\cdot \vdash \top}\ \textsc{LL\_Tr} \qquad \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\ \textsc{LL\_Tenl}$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \otimes B, \Delta, \Delta'}\ \textsc{LL\_Tenr} \qquad \frac{}{\perp \vdash \cdot}\ \textsc{LL\_Pl} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \perp, \Delta}\ \textsc{LL\_Pr}$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \parr B \vdash \Delta, \Delta'}\ \textsc{LL\_Parl} \qquad \frac{\Gamma \vdash \Delta, A, B, \Delta'}{\Gamma \vdash \Delta, A \parr B, \Delta'}\ \textsc{LL\_Parr}$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, A \multimap B, \Gamma' \vdash \Delta, \Delta'}\ \textsc{LL\_Impl} \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \multimap B, \Delta}\ \textsc{LL\_Impr}$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, B, A \vdash \Delta}\ \textsc{LL\_Exl} \qquad \frac{\Gamma \vdash \Delta_1, A, B, \Delta_2}{\Gamma \vdash \Delta_1, B, A, \Delta_2}\ \textsc{LL\_Exr}$$

Figure 3: Multi-Conclusion Linear Logic

case of FILL, so we can modify Melliès and Tabareau's translation into one that better fits the source logical system.

The following definition provides a translation of linear logic formulas into FILL formulas.

Definition 12. The following is the double-negation translation of linear logic into FILL:

$$
\begin{array}{lcl}
(\top)^N & = & \top \\
(\bot)^N & = & \bot \\
(A^\bot)^N & = & \neg((A)^N) \\
(A \parr B)^N & = & \neg\neg((A)^N) \parr \neg\neg((B)^N) \\
(A \otimes B)^N & = & \neg\neg((A)^N) \otimes \neg\neg((B)^N)
\end{array}
$$

The main point of the previous definition is that because FILL has intuitionistic versions of all of the operators of linear logic we can give a very natural translation that preserves these operators by double negating their arguments.

At this point we need to extend the translation of linear logic formulas to sequents. However, we must be careful, because in FILL implication has the FILL restriction, and thus, if we choose the wrong translation then we will run into problems trying to satisfy the FILL condition. The method we employ here is to first translate a linear logic sequent into a single-sided sequent, and then translate that to FILL using the well-known translation. That is, it is easy to see that any linear logic sequent $A_1, \ldots, A_i \vdash B_1, \ldots, B_j$ is logically equivalent to the sequent $\cdot \vdash {A_1}^\bot, \ldots, {A_i}^\bot, B_1, \ldots, B_j$. Then we translate the latter into FILL as $x_1 : \neg(({A_1}^\bot)^N), \ldots, x_i : \neg(({A_i}^\bot)^N), y_1 : \neg((B_1)^N), \ldots, y_j : \neg((B_j)^N) \vdash \cdot$ for any free variables $x_1, \ldots, x_i$ and $y_1, \ldots, y_j$, but this is equivalent to $x_1 : \neg\neg((A_1)^N), \ldots, x_i : \neg\neg((A_i)^N), y_1 : \neg((B_1)^N), \ldots, y_j : \neg((B_j)^N) \vdash \cdot$. The reader may realize that this is indeed the translation of single-sided classical linear logic into single-conclusion intuitionistic linear logic. This translation also has the benefit that we do not have to worry about mentioning terms in the statement of the result.

Lemma 9 (Negative Translation). *If $A_1, \ldots, A_i \vdash B_1, \ldots, B_j$ is derivable, then for any unique fresh variables $x_1, \ldots, x_i$, and $y_1, \ldots, y_j$, the sequent $x_1 : \neg\neg((A_1)^N), \ldots, x_i : \neg\neg((A_i)^N), y_1 : \neg((B_1)^N), \ldots, y_j : \neg((B_j)^N) \vdash \cdot$ is derivable.*

Proof. This can be shown by induction on the assumed sequent. For the complete proof see the companion report [14].

6. Agda Formalization: A Library for Dialectica Categories

The majority of the results from the last few sections have been about the category $\mathsf{Dial_2}(\mathsf{Sets})$ which, as we have said, corresponds to the dialectica category GC from de Paiva's thesis [2]. All of those results have been formalized in the proof assistant Agda[4]. This formalization is based on a general library for proving properties of dialectica categories in sets that we believe may be of interest to the wider community. This section introduces this library, but before we recall a generalization of the dialectica categories from de Paiva's thesis [2].

6.1. Lineales and Dialectica Spaces

As we introduced in Definition 4 the objects of $\mathsf{Dial_2}(\mathsf{Sets})$ are triples (U, X, α) where $\alpha \subseteq U \times X$. Equivalently, the relation α can be taken as a function $\alpha : U \times X \longrightarrow \mathsf{2}$, where $\mathsf{2} = \{0, 1\}$. Hyland and de Paiva showed that these relations can be generalized to maps of the form $\alpha : U \times X \longrightarrow L$ where the set L is a lineale [22], but here we give a slightly more general definition.

6.1.1. Lineales

A lineale is essentially a symmetric monoidal closed category in the category of prosets – preorder sets – which we call a monoidal proset.

Definition 13. A **monoidal proset** is a proset, $(L, \leq)$, with a given symmetric monoidal structure $(L, \circ, e)$. That is, a set L with a given binary relation $\leq : L \times L \longrightarrow L$ satisfying the following:

- (reflexivity) $a \leq a$ for any $a \in L$
- (transitivity) If $a \leq b$ and $b \leq c$, then $a \leq c$

together with a monoidal structure $(\circ, e)$ consisting of a binary operation, called multiplication, $\circ : L \times L \longrightarrow L$ and a distinguished element $e \in L$ called the unit such that the following hold:

- (associativity) $(a \circ b) \circ c = a \circ (b \circ c)$

[4]More about Agda can be found here `http://wiki.portal.chalmers.se/agda/pmwiki.php`, and the Agda development associated with this article can be found at `https://github.com/heades/cut-fill-agda`.

- (identity) $a \circ e = a = e \circ a$
- (symmetry) $a \circ b = b \circ a$

Finally, the structures must be compatible, that is, if $a \leq b$, then $a \circ c \leq b \circ c$ for any $c \in L$.

Now a lineale can be seen as essentially a symmetric monoidal closed category in the category prosets.

Definition 14. A **lineale** is a monoidal proset, $(L, \leq, \circ, e)$, with a given binary operation, called implication, $\multimap : L \times L \longrightarrow L$ such that the following hold:

- (relative complement) $a \circ (a \multimap b) \leq b$
- (adjunction) If $a \circ y \leq b$, then $y \leq a \multimap b$

The set $\mathsf{2}$ is an example of a lineale where the order is the usual one, the multiplication is boolean conjunction, and the implication is boolean implication. This example is not that interesting, because $\mathsf{2}$ is a boolean algebra. An example of a proper lineale can be given using the three element set $\{0, \frac{1}{2}, 1\}$ and the details – along with some other examples – can be found in the Agda development[5], but one must be careful when defining lineales, because it is possible to instead define Heyting algebras, and hence, become nonlinear.

The definition of lineales given here is a slight generalization over the original definition given by Hyland and de Paiva – see Definition 1 of [22]. They base lineales on posets instead of prosets, but the formalization given here shows that anti-symmetry can be safely dropped. In fact, the formalization we present in the next section is nearly a complete formal verification of all of their results.

6.1.2. Dialectica Spaces

Dialectica categories come in two flavors: one called GC and one called DC. The former specialized to sets where the third coordinate of the objects are binary relations corresponds to $\mathsf{Dial}_2(\mathsf{Sets})$. In this section we specialize

[5]`https://github.com/heades/dialectica-spaces/blob/master/concrete-lineales.agda#L123`

both GC and DC to sets using lineales whose objects we call **dialectica spaces**.

The category $\mathsf{Dial_L}(\mathsf{Sets})$. This category of dialectica spaces is defined similarly to the definition given in Definition 4. The following definition is due to Hyland and de Paiva [22].

Definition 15. Suppose we are given a lineale $(L, \leq, \circ, e, \multimap_L)$. Then the dialectica space $\mathsf{Dial_L}(\mathsf{Sets})$ is a category that consists of

- objects, or dialectica spaces, that are triples, $A = (U, X, \alpha)$, where U and X are sets, and $\alpha : U \times X \longrightarrow L$ is a function, and
- maps that are pairs $(f, F) : (U, X, \alpha) \longrightarrow (V, Y, \beta)$ where $f : U \longrightarrow V$ and $F : Y \longrightarrow X$ such that
  - For any $u \in U$ and $y \in Y$, $\alpha(u, F(y)) \leq \beta(f(u), y)$.

The category $\mathsf{Dial_2}(\mathsf{Sets})$ is simply an instantiation of the previous category with the concrete lineale 2.

The category $\mathsf{DC_L}(\mathsf{Sets})$. The dialectica category DC was first proposed in de Paiva's thesis [2]. It corresponds to a model of single-conclusion intuitionistic linear logic without par. We can restrict its definition to sets just as we did for the category GC.

Definition 16. Suppose we are given a lineale $(L, \leq, \circ, e, \multimap_L)$. Then the dialectica space $\mathsf{DC_L}(\mathsf{Sets})$ is a category that consists of

- objects, or dialectica spaces, that are triples, $A = (U, X, \alpha)$, where U and X are sets, and $\alpha : U \times X \longrightarrow L$ is a function, and
- maps that are pairs $(f, F) : (U, X, \alpha) \longrightarrow (V, Y, \beta)$ where $f : U \longrightarrow V$ and $F : U \times Y \longrightarrow X$ such that
  - For any $u \in U$ and $y \in Y$, $\alpha(u, F(u, y)) \leq \beta(f(u), y)$.

The differences between the two categories can be seen in the definition of morphisms between dialectica spaces. In $\mathsf{Dial_L}(\mathsf{Sets})$ the second coordinate is the map $F : Y \longrightarrow X$, but in $\mathsf{DC_L}(\mathsf{Sets})$ this map is $F : U \times Y \longrightarrow X$. The difference in the definition of morphims may seem small, but while the modification prevents the definition of a second monoidal structure modeling the connective par in the category $\mathsf{DC_2}(\mathsf{Sets})$, it is this category that has a free comonoid structure, the only non merely syntactic example in the literature, we believe.

The generalizations to lineales may seem academic, but they do have applications in concurrency. category $\mathsf{Dial}_2(\mathsf{Sets})$ shares many of the properties of Chu spaces on sets and on 2 [23]. Pratt and Gupta [24] showed that Chu spaces can be seen as a model of concurrency, but in classical linear logic, while dialectica spaces should provide a model of concurrency in full intuitionistic linear logic.

6.2. The Dialectica Space Agda Library

Proving properties about constructions on dialectica spaces can be quite involved due to the large number of nested definitions. This complexity can be seen in the proofs of Lemma 4, Lemma 7, and Lemma 8. One of the many powers of proof assistants like Agda is their ability to simplify set theoretic based definitions automatically within goals exposing the minimal goal within. This greatly simplifies working with dialectica spaces. In fact, the developments here actually show that proving properties about dialectica spaces is actually easier in a proof assistant than by hand, and with a greater confidence that the proofs are correct.

We extracted a general library for working with dialectica spaces from our formalization of the results discussed in this article. The library itself is hosted on Github and can be found at the address `https://github.com/heades/dialectica-spaces`. It consists of the definition of lineales, `lineale.agda`, several definitions of example concrete lineales, `concrete-lineales.agda`, the general definition of $\mathsf{Dial}_{\mathsf{L}}(\mathsf{Sets})$, `DialSets.agda`, and the general definition of $\mathsf{DC}_{\mathsf{L}}(\mathsf{Sets})$, `DCSets.agda`. The definitions of the latter two are parametric in the choice of lineale. The formalization of the results of this paper can be seen as an example of how to use this library to prove properties of dialectica spaces.

7. Conclusions

We first recalled the definition of full intuitionistic linear logic using the par rule proposed by Bellin but using no proof nets. We then directly proved cut-elimination for FILL in Section 3 by adapting the well-known cut-elimination procedure for classical linear logic to FILL.

In Section 4 we showed that the category $\mathsf{Dial}_2(\mathsf{Sets})$, a model of FILL, is a full LNL model by replaying the proof that linear categories are LNL models by Benton. Then in Section 5 we showed that $\mathsf{Dial}_2(\mathsf{Sets})$ is a model of full tensorial logic. The point of this exercise in categorical logic is to show that, despite linear logicians infatuation with linear negation, there is value in keeping all your connectives independent of each other. Only making them definable in terms of others, for specific applications.

Games, especially programming language games are the main motivation for Tensorial Logic and have been one of the sources of intuitions in linear logic all along. Since we are interested in the applications of tensorial logic to concurrency, we would like to see if our slightly more general framework can be applied to this task, just as well as tensorial logic.

Independently of the envisaged applications to programming, we are also interested in developing a "man in the street" game-like explanation for the finer-grained connectives of FILL, especially for par, the multiplicative disjunction. The second author has talked about games for FILL in the style of Lorenzen [25], building up on the work of Rahman [26, 27]. Rahman showed that Lorenzen games could be defined for classical linear logic [28] and discusses a sound and complete semantics in Lorenzen games for classical linear logic. Rahman suggests that one could adopt a particular structural rule enforcing intuitionism, but we have not seen a fuller discussion nor a proof of soundness and completeness for semantics of such a system. As future work we would like to show that by adapting our work on FILL we can actually obtain a sound and complete semantics of Lorenzen games.

References

[1] V. de Paiva, Dialectica categories, in: J. Gray, A. Scedrov (Eds.), Categories in Computer Science and Logic, Vol. 92, American Mathematical Society, 1989, pp. 47–62.

[2] V. de Paiva, The dialectica categories, Ph.D. thesis, University of Cambridge (1988).

[3] S. Maehara, Eine darstellung der intuitionistischen logik in der klassischen, Nagoya Mathematical Journal 7 (1954) 45–64.

[4] G. Takeuti, Proof Theory, Amsterdam: North-Holland, 1975.

[5] V. de Paiva, L. C. Pereira, A short note on intuitionistic propositional logic with multiple conclusions, MANUSCRITO - Rev. Int. Fil. 28 (2) (jul-dez 2005) 317 – 329.

[6] A. G. Dragalin, Mathematical Intuitionism. Introduction to Proof Theory, Vol. 67 of Translations of Mathematical Monographs, Amerian Mathematical Society, 1988.

[7] H. Schellinx, Some syntactical observations on linear logic, Journal of Logic and Computation 1 (4) (1991) 537–559.

[8] M. Hyland, V. de Paiva, Full intuitionistic linear logic (extended abstract), Annals of Pure and Applied Logic 64 (3) (1993) 273 – 291.

[9] G. Bierman, A note on full intuitionistic linear logic, Annals of Pure and Applied Logic 79 (3) (1996) 281 – 287.

[10] G. Bellin, Subnets of proof-nets in multiplicative linear logic with mix, Mathematical Structures in Computer Science 7 (1997) 663–669.

[11] T. Braüner, V. Paiva, A formulation of linear logic based on dependency-relations, in: M. Nielsen, W. Thomas (Eds.), Computer Science Logic, Vol. 1414 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 129–148.

[12] R. Clouston, J. Dawson, R. Gore, A. Tiu, Annotation-Free Sequent Calculi for Full Intuitionistic Linear Logic – Extended Version, ArXiv e-prints.

[13] P.-A. Melliès, Categorical Semantics of Linear Logic, 2009.

[14] H. E. III, V. de Paiva, Compainion report: Multiple conclusion intuitionistic linear logic and cut elimination, `http://metatheorem.org/papers/FILL-report.pdf`.

[15] R. Seely, Linear logic, *-autonomous categories and cofree coalgebras, in: Computer Science Logic, 1989.

[16] G. M. Bierman, On intuitionistic linear logic, Ph.D. thesis, Wolfson College, Cambridge (December 1993).

[17] P. N. Benton, A mixed linear and non-linear logic: Proofs, terms and models (preliminary report), Tech. Rep. UCAM-CL-TR-352, University of Cambridge Computer Laboratory (1994).

[18] M. E. Maietti, P. Maneggia, V. de Paiva, E. Ritter, Relating categorical semantics for intuitionistic linear logic, Applied Categorical Structures 13 (1) (2005) 1–36.

[19] J. Cockett, R. Seely, Weakly distributive categories, Journal of Pure and Applied Algebra 114 (2) (1997) 133 – 173.

[20] P.-A. Melliès, N. Tabareau, Linear continuations and duality, Tech. rep., Université Paris VII (2008).

[21] P.-A. Melliès, N. Tabareau, Resource modalities in tensor logic, Annals of Pure and Applied Logic 161 (5) (2010) 632 – 653.

[22] M. Hyland, V. de Paiva, Lineales, "O que nos faz pensar" Special number in Logic of "Cadernos do Dept. de Filosofia da PUC", Pontificial Catholic University of Rio de Janeiro (1991).

[23] V. de Paiva, Dialectica and chu constructions: Cousins?, Theory and Applications of Categories 17 (7) (2007) 127–152.

[24] V. Gupta, Chu spaces: a model of concurrency, Ph.D. thesis, Stanford University (1994).

[25] V. de Paiva, Abstract: Lorenzen games for full intuitionistic linear logic, 14th International Congress of Logic, Methodology and Philosophy of Science (CLMPS).

[26] L. Keiff, Dialogical logic, in: E. N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy, summer 2011 Edition, 2011.

[27] S. Rahman, L. Keiff, On how to be a dialogician, in: D. Vanderveken (Ed.), Logic, Thought and Action, Vol. 2 of Logic, Epistemology, and the Unity of Science, Springer Netherlands, 2005, pp. 359–408.

[28] S. Rahman, Un desafio para las teorias cognitivas de la competencia logica: los fundamentos pragmaticos de la semantica de la logica linear, Manuscrito XXV(2) (2002) 381–432.