A Mixed Linear and Non-Linear Logic: Proofs, Terms and Models (Extended Abstract)[*]

P. N. Benton[†]
University of Cambridge

Abstract

Intuitionistic linear logic regains the expressive power of intuitionistic logic through the ! ('of course') modality. Benton, Bierman, Hyland and de Paiva have given a term assignment system for ILL and an associated notion of categorical model in which the ! modality is modelled by a comonad satisfying certain extra conditions. Ordinary intuitionistic logic is then modelled in a cartesian closed category which arises as a full subcategory of the category of coalgebras for the comonad.

This paper attempts to explain the connection between ILL and IL more directly and symmetrically by giving a logic, term calculus and categorical model for a system in which the linear and non-linear worlds exist on an equal footing, with operations allowing one to pass in both directions. We start from the categorical model of ILL given by Benton, Bierman, Hyland and de Paiva and show that this is equivalent to having a symmetric monoidal adjunction between a symmetric monoidal closed category and a cartesian closed category. We then derive both a sequent calculus and a natural deduction presentation of the logic corresponding to the new notion of model.

1 Introduction

This paper concerns a variant of the intuitionistic fragment of Girard's linear logic [7]. Linear logic does not contain the structural rules of weakening and contraction, but these are reintroduced in a controlled way via a unary operator !. The rules for ! allow ordinary intuitionistic logic to be interpreted within intuitionisitic linear logic.

In [5, 4], Benton, Bierman, Hyland and de Paiva formulated a natural deduction presentation of the multiplicative/exponential fragment of ILL, together with a term calculus (extending the propositions as types analogy to linear logic) and a categorical model (a *linear category*). In that work, the multiplicative (i.e. $\otimes$,$\multimap$) part of the logic is modelled in a symmetric monoidal closed category (SMCC). That much is standard and well-understood. The ! modality is then modelled by a monoidal comonad on the SMCC which is required to satisfy certain extra (and non-trivial) conditions. These extra conditions are sufficient

[*]A considerably longer version of this paper is available as a University of Cambridge technical report [2].

[†]Author's address: University of Cambridge, Computer Laboratory, New Museums Site, Pembroke Street, Cambridge CB2 3QG, United Kingdom. Email: Nick.Benton@cl.cam.ac.uk. Research supported by a SERC Fellowship and the EU Esprit project LOMAPS.

to ensure that the category of coalgebras for the comonad contains a full subcategory which is cartesian closed and thus models the interpretation of IL in ILL.

Whilst the view that linear logic is primary and that ordinary logic is merely a part of linear logic is appealing , it is not necessarily always the best way of seeing the situation. This paper tries to present a more symmetric view of the relationship between IL and ILL and it seems worth trying to give some motivation for why this might be worth doing.

From a practical point of view, there are a number of reasons why the standard linear term calculus (LTC) of [5] might be considered unsuitable as the basis of a linear functional programming language. Firstly, linear functional programming is verbose and unnatural – whilst the LTC might well be a useful intermediate language for a compiler, it is not very appropriate as a language for everyday programming. If linearity is to be made visible to the programmer at all, it appears preferable to have some extension of a traditional non-linear language in which one could write the occasional linear function in order to deal with I/O, in-place update or whatever.

A more fundamental, problem is that, despite considerable research effort, the *precise* way in which a linear language can help with what we have deliberately referred to rather vaguely as 'I/O, in-place update or whatever' is still not clear. Most published proposals for using linear types to control or describe intensional features of functional programs are either unconvincing or use type systems which are only loosely inspired by linear logic. Systems in the last category can, pragmatically, be extremely successful; the most obvious example being the language CLEAN. The type system of CLEAN [1] incorporates a 'uniqueness' operator for (roughly) making non-linear types linear. This is in some sense dual to the ! of linear logic, which allows linear types to be treated non-linearly. Unique types in CLEAN are used to add destructive updates and I/O to the language in a referentially transparent way.

One (somewhat speculative) aim of the work described here is to provide a sound mathematical and logical basis for a type system like that of CLEAN. We are encouraged not only by the similarities between CLEAN and the calculus to be presented here (the LNL term calculus), but also by the fact that other researchers looking at practical implementations of linear languages have come up with systems which include aspects of the LNL term calculus. For example, Lincoln and Mitchell's linear variant [10] of the 'three instruction machine' divides memory into two spaces corresponding to linear and non-linear objects. Similarly, Wadler's 'active and passive' type system [14] separates linear from non-linear types. Jacobs [9] has also described how a sequent calculus inspired by CLEAN's uniqueness types may be interpreted using the linear categories of [4] under some extra simplifying assumptions.

From a more logical point of view, there has recently been much interest in Girard's system LU [8] and related systems in which the (multi)sets of formulae occuring in sequents are split into different zones. Formulae in some zones are treated classically, whilst those in other zones are treated linearly. Intuitionistic logics inspired by LU have been proposed by Plotkin [12] and by Wadler [15]. It is desirable to study the proof and model theory of such systems directly, rather than treating them as syntactic sugar for, for example, ordinary linear logic (if only to verify that it is possible to treat them as such syntactic sugar). The logic of this paper should turn out to be equivalent to a subsystem of LU, though there are some superficial differences of presentation.

From the categorical perspective, it seems natural to explore the more symmetric situation where one starts from an SMCC and a CCC with (adjoint) functors between them, rather than an SMCC with sufficient extra structure to ensure the existence of such a CCC. This is particularly true in the light of the fact that the definition of a linear category in [4] was arrived at mostly from the proof theory of linear logic, but also (and this was something of a 'hidden agenda') from a desire to have enough structure to be able to derive an appropriate CCC from the model.[1] It is also fair to say that the definition of a linear category is surprisingly complicated, so looking for simpler models, or simpler presentations of the same models, is a good idea.

The initial motivation for the present work comes from the categorical picture sketched in the previous paragraph. Once the definition has been made a little more precise, we shall show that such a situation leads to a comonad on the linear part of the model which automatically satisfies all the extra conditions required of a linear category, and thus gives a sound model of ILL including the ! operator. Furthermore, the converse holds – every linear category gives rise to such a pair of categories. Thus we have an alternative, simpler, definition of what constitutes a model for ILL. This can be seen as giving a purely category-theoretic reconstruction of !, in that a linear category (a model for ILL with !) is exactly what one obtains if one attempts directly to model an interpretation of IL in ILL without the !.

Another interesting feature of the model is that it gives rise to a strong monad on the CCC part. Thus one obtains a model not just of the lambda calculus, but of Moggi's computational lambda calculus [11].

Section 3 then looks at the logic and term calculus which are associated with our new notion of model. We formulate a sequent calculus presentation which satisfies cut elimination and then give an equivalent natural deduction system. This then gives, by the Curry-Howard correspondence, an interesting term calculus which combines the usual simply-typed lambda calculus with a linear lambda calculus. We also consider translations in both directions between this new term calculus and the linear calculus of [5].

2 The Categorical Picture

Our aim is to present a logic/terms/categories correspondence, similar to that between intuitionistic logic, simply-typed lambda calculus and cartesian closed categories, in which the categorical vertex of the triangle consists of (essentially) a cartesian closed category $\mathcal{C}$, a symmetric monoidal closed category $\mathcal{L}$ and a pair of functors $G : \mathcal{L} \to \mathcal{C}$ and $F : \mathcal{C} \to \mathcal{L}$ between them with $F \dashv G$. Intuitively, the requirement that the two functors be adjoint should be understood as saying that there is an interpretation of IL (the CCC) into ILL (the SMCC).

We will, however, need our categorical model to satisfy some extra conditions before we can have any hope of it modelling a logic or term calculus. It is necessary for the two functors and the unit and counit of the adjunction to behave well with respect to the monoidal structures of the two categories as this is used to represent the multicategorical structure implied by commas in contexts. We do not have the space to give full definitions of all the categorical concepts we shall need, but we can at least recall the broad outlines of the most important ones. The longer version of this paper [2] includes the details.

Given monoidal categories $(\mathcal{M}, \otimes, I)$ and $(\mathcal{M}', \otimes', I')$, a *monoidal functor* $F : \mathcal{M} \to \mathcal{M}'$ is a functor from $\mathcal{M}$ to $\mathcal{M}'$ equipped with a map $m_I : I' \to F(I)$ in $\mathcal{M}'$ and a

[1]This is not to say that there is anything in the model which is not justifiable in terms of the proof theory (given a proper proof-theoretic account of η-rules), but merely that, given that a translation of IL proofs into ILL proofs exists, any correct model for ILL must be able to reflect the translation semantically.

natural transformation $m_{X,Y} : F(X) \otimes' F(Y) \to F(X \otimes Y)$ which satisfy some coherence conditions. If $\mathcal{M}$ and $\mathcal{M}'$ are *symmetric* monoidal, then F is a *symmetric monoidal functor* if it is monoidal and in addition respects the twist maps σ and σ'.

If (F, m) and (G, n) are monoidal functors from an MC $\mathcal{M}$ to an MC $\mathcal{M}'$, then a *monoidal natural transformation* from (F, m) to (G, n) is a natural transformation f_X from F to G which is compatible with the comparison maps in an obvious way.

If $\mathcal{M}$ and $\mathcal{M}'$ are (symmetric) monoidal categories then a *(symmetric) monoidal adjunction* between them is an ordinary adjunction in which both of the functors are (symmetric) monoidal functors and both the unit and the counit of the adjunction are monoidal natural transformations.

Definition 1 *A linear/non-linear model (LNL model) consists of*

1. *a cartesian closed category* $(\mathcal{C}, 1, \times, \to)$*;*
2. *a symmetric monoidal closed category* $(\mathcal{L}, I, \otimes, \multimap)$ *and*
3. *a pair of symmetric monoidal functors* $(G, n) : \mathcal{L} \to \mathcal{C}$ *and* $(F, m) : \mathcal{C} \to \mathcal{L}$ *between them which form a symmetric monoidal adjunction with* $F \dashv G$*.*

We shall usually use A, B, C to range over objects of $\mathcal{L}$ and X, Y, Z for objects of $\mathcal{C}$. We write η and ε for, respectively, the unit and counit of the adjunction.

An important consequence of the definition of an LNL model is that as well as the natural transformations

$$m_{X,Y} : FX \otimes FY \to F(X \times Y)$$

$$n_{A,B} : GA \times GB \to G(A \otimes B)$$

and their nullary versions, the maps $m : I \to F1$ and $n : 1 \to GI$, we have a family of maps

$$p_{X,Y} : F(X \times Y) \to FX \otimes FY$$

given by the transpose of $n_{FX,FY} \circ \eta_X \times \eta_Y$:

$$F(X \times Y) \xrightarrow{F(\eta \times \eta)} F(GFX \times GFY) \xrightarrow{F(n)} FG(FX \otimes FY) \xrightarrow{\varepsilon} FX \otimes FY$$

and a map $p : F1 \to I$ given by $F1 \xrightarrow{Fn} FGI \xrightarrow{\varepsilon_I} I$.

Proposition 1 *In an LNL model (in fact for any monoidal adjunction), the maps* $m_{X,Y}$ *are the components of a natural isomorphism with inverses* $p_{X,Y}$ *and, furthermore, the map* m *is an isomorphism with inverse* p*:*

$$F(X) \otimes F(Y) \;\cong\; F(X \times Y)$$

$$I \;\cong\; F(1)$$

□

So F preserves the monoidal structure up to an isomorphism rather than merely up to a comparison. That is to say, F is a *strong* functor. There is, of course, a lot more interesting structure in an LNL model. To begin with, the adjunction induces a comonad on $\mathcal{L}$ and a monad on $\mathcal{C}$. We discuss each of these below.

2.1 The Comonad and Comparison with Linear Categories

The comonad on $\mathcal{L}$ is $(FG, \varepsilon : FG \to 1, \delta : FG \to FGFG)$ where ε is the counit of the adjunction and δ has components $\delta_A = F(\eta_{G(A)})$. We write ! for FG.

Lemma 2 *The comonad $(!, \varepsilon, \delta)$ is symmetric monoidal, i.e. there is a natural transformation q with components $q_{A,B} : !A \otimes !B \to !(A \otimes B)$ and a map $q : I \to !I$ such that $(!, q)$ is a symmetric monoidal functor and ε and δ are monoidal natural transformations.* □

In [4], we defined a model of the multiplicative/exponential fragment of intuitionistic linear logic as follows:

Definition 2 *A* linear category *is specified by the following data:*

1. *A symmetric monoidal closed category $(\mathcal{L}, \otimes, I, \multimap)$.*
2. *A symmetric monoidal comonad $(!, \varepsilon, \delta, q)$ on $\mathcal{L}$.*
3. *Monoidal natural transformations with components $e_A : !A \to I$ and $d_A : !A \to !A \otimes !A$ such that*
   - *(a) each $(!A, e_A, d_A)$ is a commutative comonoid,*
   - *(b) e_A and d_A are coalgebra maps, and*
   - *(c) all coalgebra maps between free coalgebras preserve the comonoid structure.*

2.1.1 Linear Categories and LNL Models are Equivalent

Any LNL model includes, by definition, part 1 of Definition 2, and we have just seen (Lemma 2) that it also satisfies part 2. Furthermore, there are plausible candidates for e_A and d_A:

$$e_A \overset{\text{def}}{=} p \circ F(*_{GA})$$

where $*_{GA}$ is the unique map from GA to the terminal object 1 of $\mathcal{C}$, and

$$d_A \overset{\text{def}}{=} p_{GA,GA} \circ F(\Delta_{GA})$$

where Δ_{GA} is the diagonal map from GA to $GA \times GA$ in $\mathcal{C}$.

Theorem 3 *For any LNL model, e and d as defined above satisfy all the conditions given in part 3 of Definition 2. In other words, any LNL model is a linear category.*

Proof. This involves checking that a fairly large collection of diagrams all commute. Although this is a lot of work, none of them are *very* difficult. Proposition 1 plays an important role in several of them. Further details may be found in [2]. □

We now sketch the proof of the converse to Theorem 3. Whilst this is largely a matter of recalling results which were proved in [4], by doing this carefully we obtain a slightly better understanding of the situation.

Assume that $\mathcal{L}$ is a linear category as in Definition 2. We need to show that this gives rise to a CCC $\mathcal{C}$ and a symmetric monoidal adjunction between $\mathcal{L}$ and $\mathcal{C}$ as in Definition 1. Recall that the comonad on $\mathcal{L}$ gives rise to two categories of algebras:

- The Eilenberg-Moore category $\mathcal{L}^!$. This has as objects all the !-coalgebras $(A, h_A : A \rightarrow !A)$ and as morphisms all the coalgebra morphisms.
- The (co-)Kleisli category $\mathcal{L}_!$. This is the full subcategory of $\mathcal{L}^!$ which has as objects all the *free* !-coalgebras $(!A, \delta_A : !A \rightarrow !!A)$.

Each of these categories comes with a pair of adjoint functors $F \dashv G$ where $G : A \mapsto (!A, \delta_A)$ and $F : (A, h_A) \mapsto A$.

Lemma 4 *If $\mathcal{L}$ is a linear category then $\mathcal{L}^!$ has finite products, with the terminal object given by $(I, q : I \rightarrow !I)$ and the product of (A, h_A) and (B, h_B) by $(A \otimes B, q_{A,B} \circ (h_A \otimes h_B))$.* □

In general, there is no reason why the Eilenberg-Moore category should be cartesian closed, since there is no reason why it should have an internal hom for arbitrary pairs of coalgebras. We can, however, find a full subcategory of the Eilenberg-Moore category which is cartesian closed.

Lemma 5 *In $\mathcal{L}^!$, all the free coalgebras are exponentiable. That is, there is an internal hom into any* free *coalgebra $(!B, \delta_B)$. Furthermore, the internal hom is itself a free coalgebra.* □

Now, notice that in any cartesian category, if an object X is exponentiable then so is $[Y, X]$ for any Y, since we can take $[Z, [Y, X]]$ to be $[Z \times Y, X]$. Furthermore, the product of two exponentiable objects X and Y is exponentiable since we can take $[Z, X \times Y]$ to be $[Z, X] \times [Z, Y]$. Taking this together with the previous lemma, we have:

Lemma 6 *The full subcategory $\mathcal{E}xp(\mathcal{L}^!)$ of the Eilenberg-Moore category having as objects the exponentiable coalgebras is cartesian closed and contains the Kleisli category $\mathcal{L}_!$.* □

Note that the Kleisli category is not, in general, itself cartesian closed, since the product of two free coalgebras is not necessarily free. We shall consider a case in which this does happen in Section 2.1.2. In the general case, we do have the following, however:

Lemma 7 *The full subcategory $\mathcal{L}_!^*$ of $\mathcal{E}xp(\mathcal{L}^!)$ consisting of finite products of free coalgebras is cartesian closed.* □

Theorem 8 *If $\mathcal{L}$ is a linear category then by taking $\mathcal{C}$ to be* either *$\mathcal{L}_!^*$* or *$\mathcal{E}xp(\mathcal{L}^!)$ and F and G to be the appropriate forgetful and free functors one obtains an LNL model.*

Proof. We have already seen that both the choices for $\mathcal{C}$ are cartesian closed so it just remains to check that F and G form a symmetric monoidal adjunction, which is straightforward. □

2.1.2 Additives and the Seely Isomorphisms

We now consider briefly what happens when an LNL model also has the extra structure required to model the additive linear connectives $\&, \oplus$ and the non-linear sum $+$. The simplest case is when the SMCC part $\mathcal{L}$ of an LNL model also has finite products, modelling the additive connective 'with' (&). The functor G preserves limits because it is a right adjoint, and in particular

$$G(A\&B) \cong GA \times GB \qquad \text{and} \qquad G1 \cong 1$$

Taking this together with Proposition 1, we obtain the following natural isomorphisms:

$$!A \otimes !B \cong !(A\&B) \qquad \text{and} \qquad I \cong !1$$

These isomorphisms were central to Seely's proposed model of ILL [13], which also proposed interpreting IL in the Kleisli category. See [6] for a critique of Seely's semantics; here we merely note the following:

Proposition 9 *If a linear category has products then the Kleisli category $\mathcal{L}_!$ is cartesian closed.*

Proof. One shows that $\mathcal{L}$ having products implies that the product of two free !-coalgebras is a free coalgebra. This means that $\mathcal{L}_!$ coincides with $\mathcal{L}_!^*$, which is cartesian closed by Lemma 7. □

The correspondence between linear categories and LNL models extends trivially to one between linear categories with finite products and LNL models with products on the SMCC part. Coproducts are slightly more problematic. Whilst the appropriate extension of an LNL model seems obvious (just require both $\mathcal{L}$ and $\mathcal{C}$ to have finite coproducts), this does not correspond quite as simply as one might hope to linear categories with coproducts.

The difficulty is that, whilst an LNL model with coproducts certainly gives rise to a linear category with coproducts, the converse does not appear necessarily to be true. Assume $\mathcal{L}$ is a linear category with finite coproducts, then $\mathcal{L}^!$ also has finite coproducts as we can define the coproduct of (A, h_A) and (B, h_B) to be

$$(A + B,\ [!\mathrm{inl} \circ h_A, !\mathrm{inr} \circ h_B])$$

and this is easily checked to satisfy the appropriate conditions. There seems no general reason, however, why either of the two CCCs which we have already identified as arising from $\mathcal{L}$ should be closed under this coproduct.

Fortunately, something can be salvaged. There are *weak* finite coproducts $\oplus$ in the Kleisli category, obtained by defining

$$(!A, \delta_A) \oplus (!B, \delta_B) \overset{\mathrm{def}}{=} (!(!A + !B), \delta_{!A+!B})$$

with, for example, the left injection given by $!\mathrm{inl} \circ \delta_A$.

2.2 The Monad and Comparison with Let-CCCs

The monad on $\mathcal{C}$ is $(GF, \eta : 1 \to GF, \mu : GFGF \to GF)$ where η is the unit of the adjunction and μ has components $\mu_X = G(\varepsilon_{FX})$. Writing T for GF, one can check that (T, η, μ) is a symmetric monoidal monad, i.e. T is a symmetric monoidal functor and both η and μ are monoidal natural transformations.

Cartesian closed categories with monoidal monads have recently been the focus of some interest, as they are the models for Moggi's computational lambda calculus [11]. The definition is usually given in terms of *strong* monads, where a monad T on a monoidal category is said to be strong if it is equipped with a natural transformation τ (called the *tensorial strength*) with components

$$\tau_{A,B} : A \otimes TB \to T(A \otimes B)$$

satsifying some extra conditions. A strong monad on a symmetric monoidal category is said to be *commutative* if the tensorial strength behaves well with respect to the twist maps σ. It turns out that commutative strong monads are the same as symmetric monoidal monads (see the full paper for more details).

A model of the computational lambda calculus (a *let-CCC*) is a cartesian closed category with a strong monad. The above implies that an LNL model always has a strong monad on the CCC part of the model and thus includes a let-CCC. The monad is, however, always commutative (because T is a *symmetric* monoidal functor). It is not the case that all strong monads on CCCs are commutative; indeed, some very important monads arising in computer science are non-commutative, for example the free monoid monad $(list, [\cdot], flatten)$ on the category of sets. Thus it is certainly the case that not all, or even all interesting, let-CCC's will arise from LNL models. Having said that, many of the most important monads arising in semantics, such as lifting and various flavours of powerset/powerdomain, *are* commutative, so the theory of commutative strong monads on CCCs is not without independent interest.

2.3 Examples

Let $\mathcal{L}$ be the category of pointed ωcpos (ω-cocomplete partial orders with a least element) and strict (bottom preserving) continuous maps. This is a symmetric monoidal closed category with tensor product given by the so-called smash product, the identity for the tensor by the one-point space (which is also a biterminator) and internal hom by the strict continuous function space. In fact, $\mathcal{L}$ also has binary products and coproducts, given by cartesian product and coalesced sum respectively.

Given this choice of $\mathcal{L}$, there are a couple of obvious choices for the CCC $\mathcal{C}$ which give an LNL model. One is to take $\mathcal{C}$ to be the category of pointed ω-cpos and continuous (not necessarily strict) maps, G to be the inclusion functor and F to be the lifting functor $F : X \to X_\perp$. The monoidal structure m on F is given by the evident isomorphism $X_\perp \otimes Y_\perp \cong (X \times Y)_\perp$. In this case, $\mathcal{C}$ is (equivalent to) the Kleisli category of the lifting comonad on $\mathcal{L}$. Note that the cartesian closure of the Kleisli category follows from the fact that $\mathcal{L}$ has products. There are strong coproducts in $\mathcal{L}$ but only weak ones in $\mathcal{C}$.

An alternative choice of $\mathcal{C}$ is the category of (not necessarily pointed) ω-cpos (these are sometimes called *predomains*) and continuous maps, again with inclusion and lifting functors. This is equivalent to the Eilenberg-Moore category of the lift comonad on $\mathcal{L}$, so it has products and coproducts by our previous general arguments, but it also turns out to be cartesian closed.

A different example arises from taking $\mathcal{L}$ be the category of Abelian groups and group homomorphisms. This is symmetric monoidal closed with $A \otimes B$ the Abelian group generated by the set of tokens $\{a \otimes b \mid a \in A, b \in B\}$ subject to the relations

$$\begin{aligned}(a_1 + a_2) \otimes b &= a_1 \otimes b + a_2 \otimes b \\ a \otimes (b_1 + b_2) &= a \otimes b_1 + a \otimes b_2\end{aligned}$$

(More categorically, $A \otimes B$ can be defined by a homomorphism $A \times B \to A \otimes B$ which is universal amongst bilinear maps into Abelian groups.) The unit for $\otimes$ is the group of integers under addition, Z, and the internal hom $A \multimap B$ is the group of homomorphisms from A to B with the multiplication inherited from B. In fact $\mathcal{L}$ also has biproducts – the direct sum $A \oplus B$ is both a product and a coproduct and the trivial group is a biterminator.

Now let $\mathcal{C}$ be $\mathcal{S}et$, and F and G be the free and forgetful functors. It is easy to check that this does indeed give an LNL model. The comonad on $\mathcal{L}$ takes an Abelian group to the free group on its underlying set. ε is 'evaluation' and η is the insertion of generators. In this case $\mathcal{C}$ is equivalent to the Kleisli category of the comonad on $\mathcal{L}$.

3 LNL Logic

LNL-models are, of course, supposed to be models of a logical system. Theorem 3 says that they are models for intuitionistic linear logic as defined by Girard, but the form of the definition of LNL-model suggests an interesting alternative presentation of the logic. The idea is that one starts with two independent logics, corresponding to the categories $\mathcal{L}$ and $\mathcal{C}$ and then adds operators which correspond in some way to the adjunction.

In keeping with our earlier conventions for naming objects of $\mathcal{L}$ and $\mathcal{C}$, we will use A, B, C to range over linear propositions and X, Y, Z for conventional ones. We shall use Γ and Δ to range over linear contexts (finite multisets of linear propositions) and Θ and Φ for non-linear ones. We also decorate turnstiles with $\mathcal{L}$ or $\mathcal{C}$ to indicate which subsystem they belong to. Finally, if Θ is $X_1, \ldots, X_n$ then $F\Theta$ means $FX_1, \ldots, FX_n$, and similarly for $G\Gamma$. The two classes of propositions with which we shall be dealing are defined by the following grammar:

$$\begin{array}{rcl} A, B & := & A_0 \mid I \mid A \otimes B \mid A \multimap B \mid FX \\ X, Y & := & X_0 \mid 1 \mid X \times Y \mid X \to Y \mid GA \end{array}$$

where A_0 (resp. X_0) ranges over some unspecified set of atomic linear (resp. non-linear) propositions.

3.1 Sequent Calculus

The two logics with which we start are very familiar *viz.* the exponential-free, multiplicative fragment of propositional intuitionistic linear logic and the $\times, \to$ fragment of ordinary intuitionistic logic. These both have very well-behaved sequent presentations. How should the systems be enriched and combined to give LNL-logicΓ There are (at least) two natural answers, neither of which satisfies cut elimination. Fortunately, there *is* a presentation of the logic which has a good proof theory. The trick is to allow conventional non-linear formulae to appear in the assumptions of a linear sequent. A typical linear sequent looks, therefore, like this:

$$X_1, \ldots, X_m, A_1, \ldots, A_n \vdash_{\mathcal{L}} B$$

which is interpreted as a morphism in $\mathcal{L}$ of the form

$$FX_1 \otimes \cdots \otimes FX_m \otimes A_1 \otimes \cdots \otimes A_n \longrightarrow B$$

Non-linear sequents are still constrained to have purely non-linear antecedents and are interpreted as morphisms in $\mathcal{C}$ in the usual way. We abuse notation by writing linear

$$A \vdash_{\mathcal{L}} A \ \mathcal{L}\textit{-axiom} \qquad X \vdash_{\mathcal{C}} X \ \mathcal{C}\textit{-axiom}$$

$$\frac{\Theta, X, X; \Gamma \vdash_{\mathcal{L}} A}{\Theta, X; \Gamma \vdash_{\mathcal{L}} A}\ \mathcal{L}\textit{-contraction} \qquad \frac{\Theta, X, X \vdash_{\mathcal{C}} Y}{\Theta, X \vdash_{\mathcal{C}} Y}\ \mathcal{C}\textit{-contraction}$$

$$\frac{\Theta; \Gamma \vdash_{\mathcal{L}} A}{\Theta, X; \Gamma \vdash_{\mathcal{L}} A}\ \mathcal{L}\textit{-weakening} \qquad \frac{\Theta \vdash_{\mathcal{C}} Y}{\Theta, X \vdash_{\mathcal{C}} Y}\ \mathcal{C}\textit{-weakening}$$

$$\frac{\Theta \vdash_{\mathcal{C}} X \qquad X, \Phi; \Gamma \vdash_{\mathcal{L}} A}{\Theta, \Phi; \Gamma \vdash_{\mathcal{L}} A}\ \mathcal{CL}\textit{-cut} \qquad \frac{\Theta \vdash_{\mathcal{C}} X \qquad X, \Phi \vdash_{\mathcal{C}} Y}{\Theta, \Phi \vdash_{\mathcal{C}} Y}\ \mathcal{CC}\textit{-cut}$$

$$\frac{\Theta; \Gamma \vdash_{\mathcal{L}} A \qquad \Phi; A, \Delta \vdash_{\mathcal{L}} B}{\Theta, \Phi; \Gamma, \Delta \vdash_{\mathcal{L}} B}\ \mathcal{LL}\textit{-cut}$$

$$\frac{\Theta, X \vdash_{\mathcal{C}} Z}{\Theta, X \times Y \vdash_{\mathcal{C}} Z}\ \mathcal{C}\text{-}\times\textit{-left1} \qquad \frac{\Theta, Y \vdash_{\mathcal{C}} Z}{\Theta, X \times Y \vdash_{\mathcal{C}} Z}\ \mathcal{C}\text{-}\times\textit{-left2}$$

$$\frac{\Theta, X; \Gamma \vdash_{\mathcal{L}} A}{\Theta, X \times Y; \Gamma \vdash_{\mathcal{L}} A}\ \mathcal{L}\text{-}\times\textit{-left1} \qquad \frac{\Theta, Y; \Gamma \vdash_{\mathcal{L}} A}{\Theta, X \times Y; \Gamma \vdash_{\mathcal{L}} A}\ \mathcal{L}\text{-}\times\textit{-left2}$$

$$\frac{\Theta \vdash_{\mathcal{C}} X \qquad \Phi \vdash_{\mathcal{C}} Y}{\Theta, \Phi \vdash_{\mathcal{C}} X \times Y}\ \times\textit{-right} \qquad \frac{}{\vdash_{\mathcal{C}} 1}\ 1\textit{-right}$$

$$\frac{\Theta; \Gamma, A, B \vdash_{\mathcal{L}} C}{\Theta; \Gamma, A \otimes B \vdash_{\mathcal{L}} C}\ \otimes\textit{-left} \qquad \frac{\Theta; \Gamma \vdash_{\mathcal{L}} A \qquad \Phi; \Delta \vdash_{\mathcal{L}} B}{\Theta, \Phi; \Gamma, \Delta \vdash_{\mathcal{L}} A \otimes B}\ \otimes\textit{-right}$$

$$\frac{\Theta; \Gamma \vdash_{\mathcal{L}} A}{\Theta; \Gamma, I \vdash_{\mathcal{L}} A}\ I\textit{-left} \qquad \frac{}{\vdash_{\mathcal{L}} I}\ I\textit{-right}$$

$$\frac{\Theta \vdash_{\mathcal{C}} X \qquad Y, \Phi \vdash_{\mathcal{C}} Z}{\Theta, X \to Y, \Phi \vdash_{\mathcal{C}} Z}\ \mathcal{C}\text{-}\!\to\!\textit{-left} \qquad \frac{\Theta \vdash_{\mathcal{C}} X \qquad Y, \Phi; \Gamma \vdash_{\mathcal{L}} A}{\Theta, X \to Y, \Phi; \Gamma \vdash_{\mathcal{L}} A}\ \mathcal{L}\text{-}\!\to\!\textit{-left}$$

$$\frac{\Theta, X \vdash_{\mathcal{C}} Y}{\Theta \vdash_{\mathcal{C}} X \to Y}\ \to\textit{-right}$$

$$\frac{\Theta; \Gamma, A \vdash_{\mathcal{L}} B}{\Theta; \Gamma \vdash_{\mathcal{L}} A \multimap B}\ \multimap\textit{-right} \qquad \frac{\Theta, \Gamma \vdash_{\mathcal{L}} A \qquad \Phi; \Delta, B \vdash_{\mathcal{L}} C}{\Theta, \Phi; \Gamma, A \multimap B, \Delta \vdash_{\mathcal{L}} C}\ \multimap\textit{-left}$$

$$\frac{\Theta \vdash_{\mathcal{C}} X}{\Theta \vdash_{\mathcal{L}} FX}\ F\textit{-right} \qquad \frac{\Theta, X; \Gamma \vdash_{\mathcal{L}} A}{\Theta; FX, \Gamma \vdash_{\mathcal{L}} A}\ F\textit{-left}$$

$$\frac{\Theta; B, \Gamma \vdash_{\mathcal{L}} A}{\Theta, GB; \Gamma \vdash_{\mathcal{L}} A}\ G\textit{-left} \qquad \frac{\Theta \vdash_{\mathcal{L}} A}{\Theta \vdash_{\mathcal{C}} GA}\ G\textit{-right}$$

Figure 1: Sequent calculus presentation of LNL logic

$$\frac{Y_1 \times \cdots \times Y_n \xrightarrow{e} X \qquad FX \otimes F\Phi \otimes \Gamma \xrightarrow{f} A}{\left(\bigotimes_i FY_i\right) \otimes F\Phi \otimes \Gamma \xrightarrow{m \otimes 1 \otimes 1} F\left(\prod_i Y_i\right) \otimes F\Phi \otimes \Gamma \xrightarrow{Fe \otimes 1 \otimes 1} FX \otimes F\Phi \otimes \Gamma \xrightarrow{f} A}\ \mathcal{CL}\textit{-cut}$$

$$\frac{X_1 \times \cdots \times X_n \xrightarrow{e} X}{FX_1 \otimes \cdots \otimes FX_n \xrightarrow{m} F(X_1 \times \cdots \times X_n) \xrightarrow{Fe} FX}\ F\textit{-right}$$

$$\frac{F\Theta \otimes B \otimes \Gamma \xrightarrow{e} A}{F\Theta \otimes FGB \otimes \Gamma \xrightarrow{1 \otimes \varepsilon \otimes 1} F\Theta \otimes B \otimes \Gamma \xrightarrow{e} A}\ G\textit{-left}$$

$$\frac{FX_1 \otimes \cdots \otimes FX_n \xrightarrow{e} A}{\prod_i X_i \xrightarrow{\eta} GF\left(\prod_i X_i\right) \xrightarrow{Gm^{-1}} G\left(\bigotimes_i FX_i\right) \xrightarrow{Ge} GA}\ G\textit{-right}$$

Figure 2: Categorical interpretation of LNL logic (sketch)

sequents in the form $\Theta; \Gamma \vdash_{\mathcal{L}} A$, even though there is no need for the ';' since linear and non-linear formulae can never be confused. Figure 1 shows the sequent rules for LNL logic.

The interpretation of LNL logic in an LNL model is fairly straightforward. We omit the interpretation of the standard logical connectives and just give details of the interpretation of one cut rule and some of the rules for F and G in Figure 2.

Theorem 10 (Cut Elimination) *There is an algorithm which, given a proof Π of a sequent $\Theta \vdash_{\mathcal{C}} X$ or $\Theta; \Gamma \vdash_{\mathcal{L}} A$, yields a cut-free proof Π' of the same sequent.*

Proof. This follows the broad outline of most cut elimination proofs, showing that proofs may be simplified by a (non-deterministic) succession of local rewrites which percolate the cuts upwards. Again, see the full version of the paper for details. $\square$

Theorem 11 *The cut elimination procedure for LNL logic is modelled soundly in any LNL model.*

Proof. One shows that whenever one proof is simplified to another then the interpretations of those two proofs are equal morphisms in the model. $\square$

There are many possible variations on the sequent rules for LNL logic. One of the most natural is to treat the non-linear parts of antecedents as additive rather than multiplicative. This yields an equivalent logic containing rules such as

$$\frac{\Theta; \Gamma \vdash_{\mathcal{L}} A \qquad \Theta; \Delta \vdash_{\mathcal{L}} B}{\Theta; \Gamma, \Delta \vdash_{\mathcal{L}} A \otimes B}\ \otimes\textit{-right}$$

One can also present the purely multiplicative version of the logic in a concise way by using some new metavariables: let P, Q range over either linear or non-linear propositions

$$\Theta; a\colon A \vdash_{\mathcal{L}} a\colon A \qquad\qquad \Theta, x\colon X \vdash_{\mathcal{C}} x\colon X$$

$$\frac{\Theta \vdash_{\mathcal{C}} s\colon X \qquad \Theta \vdash_{\mathcal{C}} t\colon Y}{\Theta \vdash_{\mathcal{C}} (s,t)\colon X \times Y} \qquad\qquad \frac{}{\Theta \vdash_{\mathcal{C}} ()\colon 1}$$

$$\frac{\Theta \vdash_{\mathcal{C}} s\colon X \times Y}{\Theta \vdash_{\mathcal{C}} \mathsf{fst}(s)\colon X} \qquad\qquad \frac{\Theta \vdash_{\mathcal{C}} s\colon X \times Y}{\Theta \vdash_{\mathcal{C}} \mathsf{snd}(s)\colon Y}$$

$$\frac{\Theta; \Gamma \vdash_{\mathcal{L}} e\colon A \qquad \Theta; \Delta \vdash_{\mathcal{L}} f\colon B}{\Theta; \Gamma, \Delta \vdash_{\mathcal{L}} e \otimes f\colon A \otimes B} \qquad \frac{\Theta; \Gamma \vdash_{\mathcal{L}} e\colon A \otimes B \qquad \Theta; \Delta, a\colon A, b\colon B \vdash_{\mathcal{L}} f\colon C}{\Theta; \Gamma, \Delta \vdash_{\mathcal{L}} \mathsf{let}\ a \otimes b = e\ \mathsf{in}\ f\colon C}$$

$$\frac{}{\Theta \vdash_{\mathcal{L}} *\colon I} \qquad\qquad \frac{\Theta; \Gamma \vdash_{\mathcal{L}} e\colon I \qquad \Theta; \Delta \vdash_{\mathcal{L}} f\colon A}{\Theta; \Gamma, \Delta \vdash_{\mathcal{L}} \mathsf{let}\ * = e\ \mathsf{in}\ f\colon A}$$

$$\frac{\Theta, x\colon X \vdash_{\mathcal{C}} s\colon Y}{\Theta \vdash_{\mathcal{C}} (\lambda x\colon X.s)\colon X \to Y} \qquad\qquad \frac{\Theta \vdash_{\mathcal{C}} s\colon X \to Y \qquad \Theta \vdash_{\mathcal{C}} t\colon X}{\Theta \vdash_{\mathcal{C}} s\, t\colon Y}$$

$$\frac{\Theta; \Gamma, a\colon A \vdash_{\mathcal{L}} e\colon B}{\Theta; \Gamma \vdash_{\mathcal{L}} (\lambda a\colon A.e)\colon A \multimap B} \qquad\qquad \frac{\Theta; \Gamma \vdash_{\mathcal{L}} e\colon A \multimap B \qquad \Theta; \Delta \vdash_{\mathcal{L}} f\colon A}{\Theta; \Gamma, \Delta \vdash_{\mathcal{L}} e\, f\colon B}$$

$$\frac{\Theta \vdash_{\mathcal{C}} s\colon X}{\Theta \vdash_{\mathcal{L}} \mathsf{F}(s)\colon FX} \qquad\qquad \frac{\Theta; \Gamma \vdash_{\mathcal{L}} e\colon FX \qquad \Theta, x\colon X; \Delta \vdash_{\mathcal{L}} f\colon A}{\Theta; \Gamma, \Delta \vdash_{\mathcal{L}} \mathsf{let}\ \mathsf{F}(x) = e\ \mathsf{in}\ f\colon A}$$

$$\frac{\Theta \vdash_{\mathcal{L}} e\colon A}{\Theta \vdash_{\mathcal{C}} \mathsf{G}(e)\colon GA} \qquad\qquad \frac{\Theta \vdash_{\mathcal{C}} s\colon GA}{\Theta \vdash_{\mathcal{L}} \mathsf{derelict}(s)\colon A}$$

Figure 3: LNL term assignment system

and Υ over mixed contexts. Then we can, for example, capture both $\to$-left rules in the one rule

$$\frac{\Theta \vdash X \qquad Y, \Upsilon \vdash P}{\Theta, X \to Y, \Upsilon \vdash P}\ \to\textit{-left}$$

This gives a set of rules which are essentially the same as those given by Jacobs in [9] (which also contains a good account of some concrete categorical models). Jacobs gives a rather different account of the semantics and there are also some subtle differences in the proof rules.

3.2 Natural Deduction and LNL Terms

There is a natural deduction formulation of LNL logic and an associated normalisation procedure. This gives, by the Curry-Howard correspondence, a term assignment system and a set of reduction rules, i.e. a mixed linear/non-linear lambda calculus. The natural deduction system we present corresponds to the additive context variant of the sequent calculus and is given in 'sequent style', complete with the term annotations, in Figure 3.

$$
\begin{aligned}
\mathsf{fst}(s,t) &\rightarrow_\beta s \\
\mathsf{snd}(s,t) &\rightarrow_\beta t \\
\mathsf{let}\ a\otimes b = e\otimes f\ \mathsf{in}\ g &\rightarrow_\beta g[e/a, f/b] \\
\mathsf{let}\ * = *\ \mathsf{in}\ e &\rightarrow_\beta e \\
(\lambda x{:}\,X.s)\ t &\rightarrow_\beta s[t/x] \\
(\lambda a{:}\,A.e)\ f &\rightarrow_\beta e[f/a] \\
\mathsf{let}\ \mathsf{F}(x) = \mathsf{F}(s)\ \mathsf{in}\ e &\rightarrow_\beta e[s/x] \\
\mathsf{derelict}(\mathsf{G}(e)) &\rightarrow_\beta e
\end{aligned}
$$

Figure 4: Term calculus β-reductions

It is easy to check that terms code derivations uniquely and that the natural deduction system is equivalent to the sequent calculus. The proof of the equivalence uses the important lemmas that substitution and weakening are admissible rules in the natural deduction system. Linear variables a,b in the context occur free exactly once in a well-typed term, whereas non-linear variables x,y may occur any number of times, including 0. Note also that there is no explicit syntax for weakening or contraction. We omit the details of the interpretation of natural deductions in LNL models.

The fundamental kind of normalisation step on natural deductions is the removal of a 'detour' in the deduction, which consists of an introduction rule immediately followed by the corresponding elimination. For reasons of space, we omit the details of the reductions on proofs but merely list the induced β-reductions on terms in Figure 4.

There is also a secondary class of reductions – the *commuting conversions*, of which there are 12 in total. The following is a typical term reduction induced by a commuting conversion:

$$\mathsf{let}\ a\otimes b = (\mathsf{let}\ * = e\ \mathsf{in}\ f)\ \mathsf{in}\ g \ \rightarrow_c\ \mathsf{let}\ * = e\ \mathsf{in}\ (\mathsf{let}\ a\otimes b = f\ \mathsf{in}\ g)$$

The reduction relation $\rightarrow_{\beta,c}$, which is the precongruence closure of the union of $\rightarrow_\beta$ and $\rightarrow_c$, is easily checked to preserve types. We also have (cf. Theorem 11):

Theorem 12 *Both the β-reductions and the commuting conversions are soundly modelled by the interpretation of the natural deduction system in any LNL model.*

- *If* $\Theta;\Gamma \vdash_{\mathcal{L}} e{:}\,A$ *and* $e \rightarrow_{\beta,c} e'$ *then* $[\![\Theta;\Gamma \vdash_{\mathcal{L}} e{:}\,A]\!] = [\![\Theta;\Gamma \vdash_{\mathcal{L}} e'{:}\,A]\!]$
- *If* $\Theta \vdash_{\mathcal{C}} s{:}\,X$ *and* $s \rightarrow_{\beta,c} s'$ *then* $[\![\Theta \vdash_{\mathcal{C}} s{:}\,X]\!] = [\![\Theta \vdash_{\mathcal{C}} s'{:}\,X]\!]$

$\square$

We can define translations in both directions between LNL logic and ILL. If A is an ILL proposition, define the linear LNL proposition A° inductively as follows:

$$
\begin{aligned}
A_0^\circ &= A_0\ (A_0 \text{ atomic}) & (A\otimes B)^\circ &= A^\circ \otimes B^\circ \\
(A \multimap B)^\circ &= A^\circ \multimap B^\circ & I^\circ &= I
\end{aligned}
$$
$$(!A)^\circ = FG(A^\circ)$$

Theorem 13 *If $\Gamma \vdash e\colon A$ in ILL, then there is an e° such that $\Gamma^\circ \vdash_{\mathcal{L}} e^\circ\colon A^\circ$.* $\square$

In the other direction, one translates the linear part of LNL logic essentially unchanged and the non-linear part using a variant of the Girard translation. E.g.:

$$\begin{array}{rcl@{\qquad\qquad}rcl}
(A \otimes B)^* & = & A^* \otimes B^* & (A \multimap B)^* & = & A^* \multimap B^* \\
(FX)^* & = & !(X^*) & (X \times Y)^* & = & !(X^*) \otimes !(Y^*) \\
(X \to Y)^* & = & !(X^*) \multimap Y^* & (GA)^* & = & A^*
\end{array}$$

Theorem 14

1. *If $\Theta \vdash_{\mathcal{C}} s\colon X$ in LNL logic, there is an LTC term s^* s.t. $!\Theta^* \vdash s^*\colon X^*$*
2. *If $\Theta; \Gamma \vdash_{\mathcal{L}} e\colon A$ in LNL logic, there is an LTC term e^* s.t. $!\Theta^*, \Gamma^* \vdash e^*\colon A^*$*

$\square$

It is easy to see that for any ILL judgement $\Gamma \vdash A$, $\Gamma^{\circ *} \vdash A^{\circ *}$ is equal to the original judgement. Thus $\Gamma \vdash A$ is provable in ILL iff $\Gamma^\circ \vdash_{\mathcal{L}} A^\circ$ is provable in LNL logic. This extends to proofs in the following way:

Theorem 15 *If $\Gamma \vdash e\colon A$ in LTC, then not only is $\Gamma \vdash e^{\circ *}\colon A$ provable, but $e \approx e^{\circ *}$ where $\approx$ is the categorical equality relation on LTC terms given in [4].* $\square$

4 Conclusions and Further Work

We have given a new and intuitively appealing characterisation of categorical models of intuitionistic linear logic. We then used this presentation of the models as the basis for defining a new logic which unifies ordinary intuitionistic logic with intuitionistic linear logic. The natural deduction presentation of the new logic then led to a mixed linear and non-linear lambda calculus. LNL logic has a natural class of categorical models and a well-behaved proof theory in both its sequent calculus and natural deduction formulations. Given this, and the links with other research which were mentioned in the introduction, LNL logic certainly seems to merit further study.

On the theoretical side, much remains to be done. We have not proved a completeness theorem, nor have we proved that the LNL term calculus is strong normalising. The strong normalisation proof should be relatively easy to do via a translation argument like that which we have previously used for the linear term calculus [3] and the computational lambda calculus. It would be nice to have better (that is, less degenerate) examples of concrete models and one might well find such examples by looking at some of the categories arising in game semantics.

We should investigate further how to treat the additives. Beyond that, one could consider adding inductive or coinductive datatypes or second-order quantification to the logic. This seems particularly worthwhile in the light of Plotkin's work on parametricity and recursion in a logic rather like ours [12].

On the practical side, we should investigate whether or not the LNL term calculus lends itself more readily to efficient implementation than does the linear term calculus. The hope is that one can arrange an implementation with two memory spaces, corresponding to the two subsystems of LNL logic. The non-linear space would be garbage collected in the usual way, whereas the linear space would contain objects satisfying some useful memory invariant (such as having only one pointer to them at all times) which could be exploited to reduce the space usage of programs. Previous experience, however, shows that turning such intuitively plausible hopes into provably correct implementations is a non-trivial task.

References

[1] E. Barendsen and S. Smetsers. Conventional and uniqueness typing in graph rewrite systems. Technical Report CSI-R9328, Katholieke Universiteit Nijmegen, December 1993.

[2] P. N. Benton. A mixed linear and non-linear logic: Proofs, terms and models (preliminary report). Technical Report 352, Computer Laboratory, University of Cambridge, September 1994.

[3] P. N. Benton. Strong normalisation for the linear term calculus. *Journal of Functional Programming*, 1995. To appear. Also available as Technical Report 305, University of Cambridge Computer Laboratory, July 1993.

[4] P. N. Benton, G. M. Bierman, J. M. E. Hyland, and V. C. V. de Paiva. Linear lambda calculus and categorical models revisited. In E. Börger et al., editor, *Selected Papers from Computer Science Logic '92*, volume 702 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[5] P. N. Benton, G. M. Bierman, J. M. E. Hyland, and V. C. V. de Paiva. A term calculus for intuitionistic linear logic. In M. Bezem and J. F. Groote, editors, *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[6] G. M. Bierman. On intuitionistic linear logic (revised version of PhD thesis). Technical Report 346, Computer Laboratory, University of Cambridge, August 1994.

[7] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[8] J.-Y. Girard. On the unity of logic. *Annals of Pure and Applied Logic*, 59:201–217, 1993.

[9] B. Jacobs. Conventional and linear types in a logic of coalgebras. Preprint, University of Utrecht, April 1993.

[10] P. Lincoln and J. C. Mitchell. Operational aspects of linear lambda calculus. In *Proceedings of the 7th Annual Symposium on Logic in Computer Science*. IEEE, 1992.

[11] E. Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[12] G. D. Plotkin. Type theory and recursion (abstract). In *Proceedings of 8th Conference on Logic in Computer Science*. IEEE Computer Society Press, 1993.

[13] R. A. G. Seely. Linear logic, *-autonomous categories and cofree coalgebras. In *Conference on Categories in Computer Science and Logic*, volume 92 of *AMS Contemporary Mathematics*, June 1980.

[14] P. Wadler. There's no substitute for linear logic (projector slides). In G. Winskel, editor, *Proceedings of the CLICS Workshop (Part I), March 1992, Aarhus, Denmark*, May 1992. Available as DAIMI PB-397-I Computer Science Department, Aarhus University.

[15] P. Wadler. A taste of linear logic. In A. M. Borzyszkowski and S. Sokolowski, editors, *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*, number 711 in Lecture Notes in Computer Science, pages 185–210, 1993.