Math. Struct. in Comp. Science (2002), *vol.* 12, *pp.* 513–539.
DOI: 10.1017/S0960129502003717 Printed in the United Kingdom

The logic of linear functors

R. BLUTE[†], J.R.B. COCKETT[‡], and R.A.G. SEELY[§]

[†] *Department of Mathematics and Statistics, University of Ottawa, 585 King Edward St, Ottawa, ON, K1N 6N5, Canada.*
Email: rblute@mathstat.uottawa.ca

[‡] *Department of Computer Science, University of Calgary, 2500 University Drive, Calgary, AL, T2N 1N4, Canada.*
Email: robin@cpsc.ucalgary.ca

[§] *Department of Mathematics and Statistics, McGill University, 805 Sherbrooke St., Montréal, QC, H3A 2K6, Canada.*
Email: rags@math.mcgill.ca

Received 24 April 2001; revised 11 November 2001

This paper describes a family of logics whose categorical semantics is based on functors with structure rather than on categories with structure. This allows the consideration of logics that contain possibly distinct logical subsystems whose interactions are mediated by functorial mappings. For example, within one unified framework, we shall be able to handle logics as diverse as modal logic, ordinary linear logic, and the 'noncommutative logic' of Abrusci and Ruet, a variant of linear logic that has both commutative and noncommutative connectives.

Although this paper will not consider in depth the categorical basis of this approach to logic, preferring instead to emphasise the syntactic novelties that it generates in the logic, we shall focus on the particular case when the logics are based on a linear functor, in order to give a definite presentation of these ideas. However, it will be clear that this approach to logic has considerable generality.

There have been several individual attempts to develop logics with distinct but related subsystems of connectives, such as the Abrusci–Ruet noncommutative logic and the bunch logic of O'Hearn and Pym; generally these are presented in terms of 'bunching' the formulas in the sequents of the logics *via* different 'punctuation'. The present functor logic, by contrast, uses a system of 'formula blocks', which represent the functorial action and give finer control over what logical features may be displayed. By displaying a family of functor logics, we illustrate how different logical systems can be developed along these lines, logics that are primarily distinguished by the degree to which they permit nesting of formula blocks. Our examples will include a basic logic where there is essentially no nesting, a system of linear modal logic, which allows nesting, but has only one system of connectives, and the

[†] Research supported in part by NSERC. Much of this work was done while this author was a guest at the University of Calgary Computer Science Department.
[‡] Research supported in part by NSERC.
[§] Research supported in part by NSERC and FCAR.

Abrusci–Ruet logic, where nesting is virtually unrestricted and there are two subsystems of connectives. We finish by showing how to translate between the 'bunch' style of logic and our 'formula block' or functor logic using the Abrusci–Ruet noncommutative logic as an example.

0. Introduction

It is quite standard to associate a logic with any categorical doctrine, and *vice versa*; in the present paper we extend this sort of correspondence to associate logics with functors between categories, an extension that allows considerable flexibility in the expressive power of the logics obtained. In particular, in this manner we can handle logics with different families of connectives, families that have some connections between them: for example, we shall be able to handle logic with both commutative and noncommutative connectives. We shall focus here on one particular setting, but the reader should keep in mind that this idea is clearly much more general.

There have been a number of attempts to develop logics with related families of connectives. One example is Abrusci and Ruet's *mixed linear logic* (Abrusci and Ruet 2000; Ruet 2000), recently renamed *noncommutative logic* or NL. The crucial characteristic of NL is that one has a commutative tensor and a (cyclic) noncommutative tensor, and that these two structures interact *via* certain structural rules. NL has proved to be an important logic that allows one to analyse computational situations in which one has both concurrent (the symmetric tensor) and sequential (the non-symmetric tensor) structures in the course of a computation. This has led, among other things, to refined proof search techniques (Maielli and Ruet 2000).

Analogously, the *bunched logic* of O'Hearn and Pym (O'Hearn and Pym 1999; O'Hearn 2000) also uses a notion of structured sequent and two families of connectives, giving in the same structure both an intuitionistic and a linear substructure; that is, the category is both cartesian and monoidal closed. In basic bunched logic, the two structures exist more or less independently of one another, but in O'Hearn (2000), O'Hearn introduced the notion of *affine bunched logic*, which allows the two logical structures to interact. This affine logic is more in the spirit of the logics in this paper.

One feature of the logics above is the use of what O'Hearn and Pym call 'bunching'. This is implemented in the syntax by having different separators in the sequents, usually comma and semicolon. So, in both NL and bunched logic, sequents typically look like $A, (B\ ; C), D \vdash E$. In contrast, we adopt a different syntax using what we call 'blocks', which contain the elements of one logical subsystem so they may act in the other subsystem. Typically, our sequents will look like $A, [B, C], D \vdash E$. The distinction between 'bunches' and 'blocks' goes beyond mere notation; the typing that goes with blocks makes for a more flexible setup, allowing for finer distinctions, some of which are illustrated by the logics in the present paper. Blocks are the embodiment of the functors underlying our logics, and the deduction rules we place on blocks reflect the properties of the functors we have in mind. The 'bunch-style' logics, such as NL and the O'Hearn–Pym bunch logic, generally have much weaker connections between the substructures, and indeed that syntax makes it difficult to impose much in the way of such connections. We shall see

that of the various block logics we present, variants that place essentially no restrictions on nesting the blocks correspond to the more familiar bunch-style logics.

As suggested above, although we think the present notions are of considerable generality, we wish to illustrate these ideas in a concrete setting. Although we shall not emphasise the categorical aspects of this work, some discussion is necessary so as to make the philosophical basis for the logics intelligible. The setting we propose to use corresponds to the tensor–par fragment of linear logic, whose categorical semantics lies in *linearly distributive categories* (Blute *et al.* 1996a; Blute *et al.* 1996b; Cockett and Seely 1997a; Cockett and Seely 1997b; Cockett and Seely 1999). Linearly distributive categories were originally introduced by the latter two authors (Cockett and Seely 1997a) with the idea that the tensor–par fragment of linear logic (Girard 1987) was crucial categorically, and that the remaining structure associated with linear logic could then be added to this basic fragment in a modular fashion. A linearly distributive category is a category equipped with two monoidal structures, related by a linear distribution. In the present context, it is desirable to allow the tensor and par to be noncommutative. This involves issues that were originally considered in Blute *et al.* (1996a), but have been studied in further detail by Schneck (Schneck 1998). Non-symmetric monoidal categories are becoming more and more important due to their many connections to physics. Of particular interest are categories of representations of Hopf algebras (Majid 1995). By having logical structures to analyse such categories and the functors between them, it is hoped to gain new insight into these examples.

Central to the current paper is the notion of a linearly distributive functor between linearly distributive categories (Cockett and Seely 1999). Rather than merely using the evident notion of a structure-preserving functor, this is instead the appropriate generalisation of the notion of a monoidal functor. So a linearly distributive functor (hereafter called simply a *linear functor*) is actually a pair of functors, one of which is monoidal with respect to the tensor and the other comonoidal with respect to par, and the pair satisfy several further coherence conditions. That the correct notion of morphism is captured follows from the observation that when the two categories are in fact $*$-autonomous, this notion reduces to having a single monoidal functor. Furthermore, an appropriate proof net system, *functor circuits*, is introduced in Cockett and Seely (1999). This calculus was inspired by previous work on the categorical structure of the exponentials **!** and **?** (Blute *et al.* 1996b). It should be emphasised that familiar linear logic constructions, such as the additives and exponentials, fit naturally into this framework. We would also note that the *topological quantum field theories* introduced by Atiyah are examples of linear functors (Atiyah 1990; Quinn 1995), thus providing a possible application to theoretical physics.

In the present paper we define a sequent calculus associated to a linear functor in such a way that any linear functor provides a model. Not only is this logic of a linear functor a natural extension of the logic of a linearly distributive category, but interest in this logic is validated by some naturally occurring examples. The novelty here is that by associating a logic to a functor rather than a category, one can have a logical system containing two distinct logical subsystems that are allowed to interact in a way mediated by the linear functor. This idea is clearly much more general than the present paper might suggest.

Some extensions of the notions developed here might include basing the logic on a family of linear functors, or even using other notions of functor.

A traditional logic that fits into the present context very well is modal logic; we shall discuss how a linear modal logic is a natural special case of our most basic logic. In this case there is only one family of tensor structures, and the variation is entirely carried by the linear functor. The other examples we shall consider place a greater emphasis on having more than one tensor family.

An illuminating example occurs in Retoré (1997). Retoré introduced a certain non-commutative tensor operator on the category of coherence spaces; the point about his example is that the identity functor is then a linear functor from the category of coherence spaces with this new monoidal (and hence degenerately linearly distributive) structure to the category of coherence spaces with its usual $*$-autonomous structure. Thus our general logic, in a specific instantiation, gives an alternate logical structure for Retoré's work. Related to this example is Abrusci and Ruet's NL, which can also be interpreted as an instance of the logic of a linear functor. O'Hearn and Pym's affine bunched logic fits into a similar framework, although its essential 'non-linear' nature makes it unsuitable for the exact context of the present paper.

We shall use our notion of 'block' logic (based on linear functors) to introduce a series of logics arising from linear functors, each having a rather different structure determined by the extent and nature of the nesting of blocks allowed. We will present sequent calculi for all the logics, present their cut-elimination theorems, and examine the models discussed above in the light of these logics. In particular, we show how to interpret NL in our functor logic, and *vice versa.*

This paper is presented in such a manner that it may be regarded as essentially self-contained. From that point of view, it is a paper that develops a family of logics, and shows connections with other well-known logics. However, this only considers part of the story; although we shall not dwell here on the categorical semantics, they may be reconstructed from Cockett and Seely (1999). In the sense that a complete understanding of the matter presented here truly involves understanding the categorical setting and the circuit diagrams (or proof nets) that we use to represent morphisms in the free categories, this paper is heavily dependent on Cockett and Seely (1999). We encourage the reader to consult that paper, which will make the categorical commentary more meaningful. Finally, the reader interested in even greater generality ought to consult Cockett *et al.* (2000), where the categorical theory (in particular, the notion of linear functor) is extended to the 2-dimensional setting of linear bicategories. Since this extra generality has no effect on the examples considered in this paper, we shall not invoke it further here.

1. The basic sequent calculus LF and its interpretation

We shall develop two types of logic arising from linear functors, each having a rather distinct structure. These distinctions will correspond logically to different notions of nesting. The first logic, which is introduced in this section, corresponds to the situation of a general linear functor between two distinct categories. This corresponds in the logical world to having two distinct logics with a weak 'interpretation' between them. This gives

rise to a relatively simple 'block' sequent calculus syntax in which there is no recursive nesting, that is, only one level of nesting is allowed in the logic that corresponds to the codomain of the linear functor.

This section describes this logic and its categorical semantics. We show that the logic has a cut elimination process. The techniques developed in this section will be the basis for the development of logics with nested block structures, to be discussed in the next section.

1.1. *The logic* LF

As is traditional in logic, we shall construct the basic logic LF from atomic formulas and non-logical axioms between these formulae. However, we see no reason why this atomic structure should not itself be closed under cut. Since we have in mind a categorical semantics in which the notion of equivalence of proofs is central, it is natural to assume that this atomic structure could have proof equivalences and so could form a category, in fact two categories. The more traditionally inclined logicians may suppose these are just sets, or graphs if nonlogical axioms are wanted. Each of these atomic structures will generate a tensor–par fragment of linear logic, and, in addition, we wish to construct a way for one logic to be interpreted in the other, both at the sequent level and as formulas. This latter wish amounts to a form of 'representability' of the blocks as explicit formulas, upon which we shall rely fairly heavily in working with this logic and its variants later in the paper.

So we begin by supposing that we have two categories $\mathbf{A}$ and $\mathbf{X}$. We shall construct, in effect, the free linearly distributive categories generated by these, and a linear functor between the generated categories $F\colon \tilde{\mathbf{A}} \longrightarrow \tilde{\mathbf{X}}$. We construct these categories by the usual logical construction (Blute *et al.* 1996a) of building the tensor–par fragment of linear logic over the generators given by the base categories. Note that we neither assume nor exclude the exchange rule. This means that we allow either, both or neither of the constructed categories to be commutative, depending on whether the exchange rule is assumed to be present or not. We shall denote the two entailment relations (for the two free categories) by $\vdash_S$ and $\vdash_T$ (for source and target). There is a new feature we must add here: in order to construct the linear functor F, we must add appropriate rules to include the logical structure from $\tilde{\mathbf{A}}$ within the T-entailment relation. Sequents in this enriched T-logic will look like the following[†]:

$$X_1, [A_1, A_2], X_2, [A_3] \vdash_T X_3, [A_4, A_5]$$

Here we use the brackets to indicate that we are including formulas from the logic associated to $\mathbf{A}$ within the T-entailment relation. One should think of the $\mathbf{A}$ (or S)

[†] We shall usually denote objects from $\tilde{\mathbf{A}}$ by letters $A, B, \ldots$ from early in the alphabet, and objects from $\tilde{\mathbf{X}}$ by letters $X, Y, \ldots$ from the end. Finite sequences of such objects will be similarly denoted, for example, Γ, Δ from $\tilde{\mathbf{A}}$, Ξ, Υ from $\tilde{\mathbf{X}}$. When either $\tilde{\mathbf{A}}$ or $\tilde{\mathbf{X}}$ may be involved, we shall use letters from the middle of the alphabet, such as M, N and Φ, Ψ.

formulas as being *nested* within the T-logic. There are new sequent calculus rules to determine how the nested S-logic interacts with the T-logic.

Although we shall not develop the details in this paper, the reader should note that this setting can be generalised to allow several linear functors between different logical settings.

We begin by defining the notions of formula and sequent. To denote the linearly distributive structures of $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{X}}$, we shall use $\otimes_S, \otimes_T, \oplus_S, \oplus_T$ for the tensors and cotensors (par's), and use $\top_S, \top_T, \bot_S, \bot_T$ for the units, dropping the subscripts when the typing context makes it clear which structure is involved.

There will be two types of sequent: S-sequents and T-sequents. S-sequents are as usual for this fragment of linear logic, see, for example, Cockett and Seely (1997a). However, T-sequents require a new notion, *blocks*, which will recur throughout the paper. For the moment, a source block is just a finite (possibly empty) sequence of S-formulas, but target blocks admit an additional construction: if Γ is an S-block, then $[\Gamma]$ is a T-block. Then S- or T-sequents may be defined in terms of S- or T-blocks.

Definition 1.1 (Formulas and Sequents). *Formulas* for the source logic (S-formulas) are the linear types constructed from the objects of $\mathbf{A}$ (the atomic formulas). Explicitly, S-formulas include the objects of $\mathbf{A}$, the tensor and par units $\top$ and $\bot$, and they are closed under tensor ($\otimes$) and cotensor (or par) ($\oplus$). The target formulas (T-formulas), in addition to being given by the objects of $\mathbf{X}$ closed under linear type constructors, have two additional *type constructors*:

— If A is a source formula, then $F_{\otimes}(A)$ and $F_{\oplus}(A)$ are target formulas.

An S-block is just a finite sequence of S-formulas.
T-blocks are defined by the following:

— If X is a T-formula, then X is a T-block. The empty sequence is a T-block.
— If Ξ_1 and Ξ_2 are T-blocks, then so is Ξ_1, Ξ_2.
— If Γ is an S-block, then $[\Gamma]$ is a T-block.

Then an *x-sequent* (for $x = S$ or T) is defined to be $\Phi \vdash_x \Psi$, where Φ and Ψ are x-blocks.

S-sequents are the usual sequents for the tensor–par fragment of linear logic; examples of typical T-sequents are:

— $X_1, [A_1, A_2], X_2, [A_3] \vdash_T X_3, [A_4, A_5], X_4 \otimes_T F_{\oplus}(A_6)$
— $\Xi, [A_1 \otimes_S A_2, A_3], \Xi' \vdash_T \Upsilon, F_{\otimes}(A_4)$.

We will now present the sequent calculus inference rules for LF. In addition to the usual linear rules for $\vdash_S$ and $\vdash_T$, we have additional rules that allow us to use the monoidal structure of the source logic while its formulas are nested in the target. The semantic intention will perhaps be clearest if we first present these rules as 'two-way' rules, by which we mean that they can be applied in either direction, and thus represent bijections between derivations of the appropriate sequents:

$$\frac{\Xi, [\Gamma, A_1, A_2, \Gamma'], \Xi' \vdash_T \Upsilon}{\Xi, [\Gamma, A_1 \otimes A_2, \Gamma'], \Xi' \vdash_T \Upsilon} \qquad \frac{\Xi, [\Gamma, \Gamma'], \Xi' \vdash_T \Upsilon}{\Xi, [\Gamma, \top, \Gamma'], \Xi' \vdash_T \Upsilon}$$

We also have the dual rules for the par structure on the right-hand side. Next we have the two-way *typing* rules:

$$\frac{\Xi, [A], \Xi' \vdash_T \Upsilon}{\Xi, F_{\otimes}(A), \Xi' \vdash_T \Upsilon}$$

and its dual for $F_{\oplus}$ on the right. These typing rules are only assumed in the case where the bracket contains a single formula, although there are evident derived rules when there is an arbitrary block in the bracket.

The categorical interpretation of these rules is that $[A_1, A_2, \ldots, A_n]$ is interpreted as $F_{\otimes}(A_1 \otimes A_2 \cdots \otimes A_n)$ on the left-hand side and as $F_{\oplus}(A_1 \oplus A_2 \oplus \cdots \oplus A_n)$ when appearing on the right.

There are (one way) *monoidal rules* that assert the monoidalness (and comonoidalness) of the two components of the linear functor:

$$\frac{\Xi, [\Gamma, \Gamma'], \Xi' \vdash_T \Upsilon}{\Xi, [\Gamma], [\Gamma'], \Xi' \vdash_T \Upsilon} \qquad \frac{\Xi, [\top], \Xi' \vdash_T \Upsilon}{\Xi, \Xi' \vdash_T \Upsilon}$$

together with the dual rules for the right-hand side. These rules are also called the *entropy rules*, a terminology inspired by the noncommutative logic of Ruet (Ruet 2000).

Of course, using two way rules like this means there is no hope of having a cut elimination result for the logic, so in our formal presentation of the sequent calculus we shall present these rules in a more conventional format, in terms of left- and right-introduction rules. In the presence of the cut rules, these will be equivalent to the two way rules above.

But before we do that, we must consider what forms of cut are permissible. Of course, both source and target logics include the usual (noncommutative) tensor–par fragment of linear logic, so there will be the usual four planar cut rules for each of $\vdash_S$ and $\vdash_T$ (Cockett and Seely 1997a). An example would be the rule

$$\frac{\Xi \vdash_T \Upsilon, X \quad X, \Xi' \vdash_T \Upsilon'}{\Xi, \Xi' \vdash_T \Upsilon, \Upsilon'}$$

But in addition we might expect cut rules that allow for the interaction of the two logical structures. These are the so-called *mixed cut rules*. Here are two examples:

$$\frac{\Gamma \vdash_S \Delta, A \quad [A, \Gamma'], \Xi \vdash_T \Upsilon}{[\Gamma, \Gamma'], \Xi \vdash_T [\Delta], \Upsilon} \qquad \frac{\Xi \vdash_T \Upsilon, [\Gamma', A] \quad A, \Gamma \vdash_S \Delta}{\Xi, [\Gamma] \vdash_T \Upsilon, [\Gamma', \Delta]}$$

Of course, there are a number of symmetric variants, which are left to the reader. However, it turns out that once we establish the appropriate left and right introduction rules for the two way rules above, in particular for the typing rules, these mixed cut rules will be derivable, and so need not be added to the sequent calculus.

In Table 1 we list the sequent rules for this logic LF, using the convention that when a rule applies to either sequent calculus, we use a variable such as x to represent either of S or T. We have given the left and right introduction rules corresponding to the two way rules discussed above. It is a standard exercise to show that in the presence of cut these are equivalent to the two way rules. We shall illustrate this with the following pair of lemmas.

Table 1. *Sequent rules for* LF

Note that x may represent either S or T.

$$\frac{A \xrightarrow{f} B \text{ in } \mathbf{A}}{A \vdash_S B}\ (f\mathbf{A}) \qquad \frac{X \longrightarrow Y \text{ in } \mathbf{X}}{X \vdash_T Y}\ (f\mathbf{X})$$

$$\frac{\Phi, M, N, \Phi' \vdash_x \Psi}{\Phi, M \otimes_x N, \Phi' \vdash_x \Psi}\ (\otimes_x L) \qquad \frac{\Phi_1 \vdash_x \Psi_1, M, \Psi_1' \quad \Phi_2 \vdash_x \Psi_2, N, \Psi_2'}{\Phi_1, \Phi_2 \vdash_x \Psi_1, \Psi_2, M \otimes_x N, \Psi_1', \Psi_2'}\ (\otimes_x R)$$

where either Ψ_1', Ψ_2 are empty, or Φ_1, Ψ_1' are empty, or Φ_2, Ψ_2 are empty.

$$\frac{\Phi_1, M, \Phi_1' \vdash_x \Psi_1 \quad \Phi_2, N, \Phi_2' \vdash_x \Psi_2}{\Phi_1, \Phi_2, M \oplus_x N, \Phi_1', \Phi_2' \vdash_x \Psi_1, \Psi_2}\ (\oplus_x L) \qquad \frac{\Phi \vdash_x \Psi, M, N, \Psi'}{\Phi \vdash_x \Psi, M \oplus_x N, \Psi'}\ (\oplus_x R)$$

where either Φ_1', Φ_2 are empty, or Φ_2, Ψ_2 are empty, or Φ_1', Ψ_1 are empty.

$$\frac{\Phi, \Phi' \vdash_x \Psi}{\Phi, \top_x, \Phi' \vdash_x \Psi}\ (\top_x L) \qquad \frac{}{\vdash_x \top_x}\ (\top_x R)$$

$$\frac{}{\bot_x \vdash_x}\ (\bot_x L) \qquad \frac{\Phi \vdash_x \Psi, \Psi'}{\Phi \vdash_x \Psi, \bot_x, \Psi'}\ (\bot_x R)$$

$$\frac{\Phi_1 \vdash_x \Psi_1, M, \Psi_1' \quad \Phi_2, M, \Phi_2' \vdash_x \Psi_2}{\Phi_2, \Phi_1 \vdash_x \Psi_2, \Psi_1}\ (cut_x)$$

where one of Φ_2, Ψ_1 is empty, and one of Φ_2', Ψ_1' is empty.

$$\frac{\Xi, [\Gamma, A_1, A_2, \Gamma'], \Xi' \vdash_T \Upsilon}{\Xi, [\Gamma, A_1 \otimes_S A_2, \Gamma'], \Xi' \vdash_T \Upsilon}\ ([\otimes]L) \qquad \frac{\Xi \vdash_T \Upsilon, [\Delta, A_1, A_2, \Delta'], \Upsilon'}{\Xi \vdash_T \Upsilon, [\Delta, A_1 \oplus_S A_2, \Delta'], \Upsilon'}\ ([\oplus]R)$$

$$\frac{\Xi, [\Gamma, \Gamma'], \Xi' \vdash_T \Upsilon}{\Xi, [\Gamma, \top_S, \Gamma'], \Xi' \vdash_T \Upsilon}\ ([\top]L) \qquad \frac{\Xi \vdash_T \Upsilon, [\Delta, \Delta'], \Upsilon'}{\Xi \vdash_T \Upsilon, [\Delta, \bot_S, \Delta'], \Upsilon'}\ ([\bot]R)$$

$$\frac{\Xi, [A], \Xi' \vdash_T \Upsilon}{\Xi, F_\otimes(A), \Xi' \vdash_T \Upsilon}\ (F_\otimes L) \qquad \frac{\Gamma \vdash_S \Delta, A, \Delta'}{[\Gamma] \vdash_T [\Delta], F_\otimes(A), [\Delta']}\ (F_\otimes R)$$

$$\frac{\Gamma, A, \Gamma' \vdash_S \Delta}{[\Gamma], F_\oplus(A), [\Gamma'] \vdash_T [\Delta]}\ (F_\oplus L) \qquad \frac{\Xi \vdash_T \Upsilon, [A], \Upsilon'}{\Xi \vdash_T \Upsilon, F_\oplus(A), \Upsilon'}\ (F_\oplus R)$$

$$\frac{\Xi, [\Gamma, \Gamma'], \Xi' \vdash_T \Upsilon}{\Xi, [\Gamma], [\Gamma'], \Xi' \vdash_T \Upsilon}\ (Mon_2 L) \qquad \frac{\Xi \vdash_T \Upsilon, [\Gamma, \Gamma'], \Upsilon'}{\Xi \vdash_T \Upsilon, [\Gamma], [\Gamma'], \Upsilon'}\ (Mon_2 R)$$

$$\frac{\Xi, [\top_S], \Xi' \vdash_T \Upsilon}{\Xi, \Xi' \vdash_T \Upsilon}\ (Mon_0 L) \qquad \frac{\Xi \vdash_T \Upsilon, [\bot_S], \Upsilon'}{\Xi \vdash_T \Upsilon, \Upsilon'}\ (Mon_0 R)$$

Lemma 1.2. In LF (as presented in Table 1), the mixed cut rules are derivable rules.

Proof. Two simple examples will illustrate the proof. The following mixed cut may be given by the derivation on the right.

$$\frac{\Gamma \vdash_S \Delta, A \quad [A], \Xi \vdash_T \Upsilon}{[\Gamma], \Xi \vdash_T [\Delta], \Upsilon} \quad = \quad \frac{\dfrac{\Gamma \vdash_S \Delta, A}{[\Gamma] \vdash_T [\Delta], F_\otimes(A)}\ (F_\otimes R) \quad \dfrac{[A], \Xi \vdash_T \Upsilon}{F_\otimes(A), \Xi \vdash_T \Upsilon}\ (F_\otimes L)}{[\Gamma], \Xi \vdash_T [\Delta], \Upsilon}\ (cut_T)$$

Next, the mixed cut

$$\frac{B \vdash_S A \quad X, [C, A, D], X' \vdash_T Y}{X, [C, B, D], X' \vdash_T Y}$$

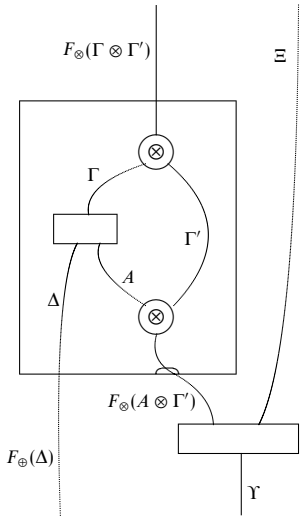


Fig. 1. Circuit for a mixed cut

proceeds *via* some canonical uses of tensor introduction:

$$\frac{B \vdash_S A}{C, B, D \vdash_S C \otimes A \otimes D} \qquad \frac{X, [C, A, D], X' \vdash_T Y}{X, [C \otimes A \otimes D], X' \vdash_T Y}$$

Note that the first is entirely in S, and the second only uses $([\otimes]L)$. Using these derived rules, the mixed cut looks like this:

$$\dfrac{\dfrac{\dfrac{B \vdash_S A}{C, B, D \vdash_S C \otimes A \otimes D}}{[C, B, D] \vdash_T F_\otimes(C \otimes A \otimes D)}\,(F_\otimes R) \quad \dfrac{\dfrac{X, [C, A, D], X' \vdash_T Y}{X, [C \otimes A \otimes D], X' \vdash_T Y}}{X, F_\otimes(C \otimes A \otimes D), X' \vdash_T Y}\,(F_\otimes L)}{X, [C, B, D], X' \vdash_T Y}\,(cut_T)$$

The general proof can be constructed from the principals illustrated above, in effect 'preparing' the sequents so that the F rules and a T-cut can simulate the mixed cut. As an example of how the circuit calculus of Cockett and Seely (1999) may be used for LF, the following mixed cut is represented by the circuit given in Figure 1, although we have simplified the figure somewhat by representing $\Gamma, \Gamma', \Delta, \Xi, \Upsilon$ by single formulas:

$$\frac{\Gamma \vdash_S \Delta, A \quad [A, \Gamma'], \Xi \vdash_T \Upsilon}{[\Gamma, \Gamma'], \Xi \vdash_T [\Delta], \Upsilon}$$

It is clear from this circuit how the mixed cut is created from the linear functor and the ordinary cut.

□

Lemma 1.3. In LF (as presented in Table 1), the two way tensor and the two way typing rules are derivable rules.

Proof. In each case, one direction is obvious. For the 'reverse' tensor rule, one merely

has to use a mixed cut on the easily derivable sequent $M, N \vdash_x M \otimes_x N$. The 'reverse' typing rule is similarly derived *via* a mixed cut on the sequent $[A] \vdash_T F_\otimes(A)$, which is a simple consequence of $(F_\otimes R)$. □

We have seen some examples of derivations in this calculus; the following examples also illustrate parts of the steps necessary for the development of the categorical semantics. We first give a derivation of the sequent $[A \oplus B] \vdash_T F_\otimes(A), [B]$, which would be interpreted categorically as the linear strength $F_\otimes(A \oplus B) \longrightarrow F_\otimes(A) \oplus F_\oplus(B)$ (for simplicity we use the derived mixed cut rule):

$$\frac{A \oplus B \vdash_S A, B \qquad \dfrac{F_\otimes(A) \vdash_T F_\otimes(A)}{[A] \vdash_T F_\otimes(A)}}{[A \oplus B] \vdash_T F_\otimes(A), [B]}$$

For another example, consider the monoidal map $F_\otimes(A) \otimes F_\otimes(B) \longrightarrow F_\otimes(A \otimes B)$. Here is a proof of the corresponding sequent $[A], [B] \vdash_T F_\otimes(A \otimes B)$:

$$\frac{\dfrac{A, B \vdash_S A \otimes B \qquad [A \otimes B] \vdash_T F_\otimes(A \otimes B)}{[A, B] \vdash_T F_\otimes(A \otimes B)}}{[A], [B] \vdash_T F_\otimes(A \otimes B)}$$

As a reminder, we note that the blocks on either side of a turnstile have different interpretations (as the two components of a linear functor), so, for example, a sequent $[A] \vdash_T [A]$ is not only not an identity axiom, it is not even derivable in LF. This would correspond to $F_\otimes(A) \longrightarrow F_\oplus(A)$, which is not in general derivable. In some situations it might be a desirable additional axiom – for instance, if F is the tensor–par linear functor, such an axiom would correspond to the mix rule. In modal logic, it would say necessity entails possibility.

1.2. *Categorical semantics*

We do not wish in this paper to belabour the categorical issues underlying the logics we are presenting, but we think it would be remiss of us not to provide some details of this semantics. The reader should certainly have gathered that LF has a sound and complete semantics in terms of linear functors. In fact, it should be evident that the logic was actually arrived at by reverse engineering the categorical notions. Of course, this would not be remarkable except for the fact that these settings had already arisen independently employing rather different logical tools, as will be clear from the examples we present in this paper.

We remind the reader of the basic definitions of linearly distributive category and linear functor; for details the reader should see the original papers, Cockett and Seely (1997a) Blute *et al.* (1996a), and Cockett and Seely (1999). A linearly distributive category consists of a category $\mathbf{C}$ together with two monoidal structures, denoted $(\mathbf{C}, \otimes, \top)$ and $(\mathbf{C}, \oplus, \bot)$, respectively. The category $\mathbf{C}$ must also be equipped with two 'linear distributions':

$$\delta_L : A \otimes (B \oplus C) \longrightarrow (A \otimes B) \oplus C$$

$$\delta_R : (B \oplus C) \otimes A \longrightarrow B \oplus (C \otimes A).$$

These are, of course, subject to a number of coherence conditions (Cockett and Seely 1997a), which make each tensor strong (or costrong) with respect to the other. We point out that any $*$-autonomous category is linearly distributive, taking the tensor and par as the two monoidal structures. Also any monoidal category is linearly distributive, taking the tensor for both monoidal structures.

Definition 1.4. Suppose now that **X** and **Y** are linearly distributive categories. A *linear functor* $F : \mathbf{X} \longrightarrow \mathbf{Y}$ consists of a pair of functors $F_\otimes, F_\oplus : \mathbf{X} \longrightarrow \mathbf{Y}$ such that:

— $F_\otimes$ is monoidal with respect to tensor. So, in particular, there are maps $m_\otimes : F_\otimes(A) \otimes F_\otimes(B) \longrightarrow F_\otimes(A \otimes B)$ and $m_\top : \top \longrightarrow F_\otimes(\top)$.
— $F_\oplus$ is comonoidal with respect to par. So, in particular, there are maps $n_\oplus : F_\oplus(A \oplus B) \longrightarrow F_\oplus(A) \oplus F_\oplus(B)$ and $n_\bot : F_\oplus(\bot) \longrightarrow \bot$.
— There must exist the following *linear strengths*:
  – $v^R_\otimes : F_\otimes(A \oplus B) \longrightarrow F_\oplus(A) \oplus F_\otimes(B)$
  – $v^L_\otimes : F_\otimes(A \oplus B) \longrightarrow F_\otimes(A) \oplus F_\oplus(B)$
  – $v^R_\oplus : F_\otimes(A) \otimes F_\oplus(B) \longrightarrow F_\oplus(A \otimes B)$
  – $v^L_\oplus : F_\oplus(A) \otimes F_\otimes(B) \longrightarrow F_\oplus(A \otimes B)$.

These must satisfy a number of coherence conditions, which are spelled out in the original paper, Cockett and Seely (1999).

As we have already pointed out, the exponential and additive operators of linear logic have natural interpretations as linear functors. Also, any monoidal functor between $*$-autonomous categories induces a linear functor.

One important example arises from work of Retoré (Retoré 1997). This example exhibits an important point: the functor parts of a linear functor may be trivial, that is, the identity on objects and morphisms, so that the structure is carried by the linear strengths alone.

Let **Coh** be the $*$-autonomous category of coherence spaces, with the usual $\otimes, \oplus$ (Girard 1987). (So objects are pairs $A = \langle S_A, R_A \rangle$ consisting of a set S_A and a symmetric, anti-reflexive relation R_A.) Define another 'lexicographic' tensor structure (called 'before') on **Coh** as follows:

$$A \leqslant B = \langle S_A \times S_B, \{ \langle (a,b),(a',b') \rangle : bR_Bb' \text{ or } b = b' \wedge aR_Aa' \} \rangle.$$

Then one may show that the identity functor is a linear functor to **Coh** with the linear structure $(\otimes, \oplus)$ from **Coh** with the linear structure given by $\leqslant$ viewing the monoidal category (**Coh**,$\leqslant$) as a 'degenerate' linearly distributive category with tensor and par coinciding:

$$\mathrm{Id} : \mathbf{Coh}_{(\leqslant,\leqslant)} \longrightarrow \mathbf{Coh}_{(\otimes,\oplus)}$$

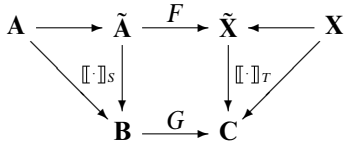
It will be clear then that our general notion of the logic of a linear functor will, in particular, give a logical interpretation to Retoré's work.

We can now return to consider the structure inherent in LF. Note that the formulas of S are the objects of the free linearly distributive category $\tilde{\mathbf{A}}$ generated by **A**, and its morphisms are represented by the derivable sequents $A \vdash_S B$, and, similarly, for T and

the free linearly distributive category $\tilde{\mathbf{X}}$ generated by $\mathbf{X}$. The assignments $A \mapsto F_{\otimes}(A)$ and $A \mapsto F_{\oplus}(A)$ are the object parts of functors $F_{\otimes}$ and $F_{\oplus}$, which make up a linear functor $F : \tilde{\mathbf{A}} \longrightarrow \tilde{\mathbf{X}}$. There are canonical embeddings of the atomic structure into the complete logic, *viz*. functors $\mathbf{A} \longrightarrow \tilde{\mathbf{A}}$ and $\mathbf{X} \longrightarrow \tilde{\mathbf{X}}$. And $F : \tilde{\mathbf{A}} \longrightarrow \tilde{\mathbf{X}}$ is the universal such entity over $\mathbf{A}$ and $\mathbf{X}$. Given any other such structure, that is, a linear functor $G : \mathbf{B} \longrightarrow \mathbf{C}$ and functors $\mathbf{A} \longrightarrow \mathbf{B}$ and $\mathbf{X} \longrightarrow \mathbf{C}$, there is a lifting to $F : \tilde{\mathbf{A}} \longrightarrow \tilde{\mathbf{X}}$ so that the evident diagram commutes. This can be expressed in logical terms in terms of the notion of an 'interpretation'; an interpretation, essentially, assigns to atomic data some appropriate data in the target categories, and it is straightforward to extend this to the complete logic, that is, to the free linearly distributive categories.

Definition 1.5. Suppose LF is given as defined above (in terms of atomic structure given by the categories $\mathbf{A}$, $\mathbf{X}$). Let $\mathbf{B}$ and $\mathbf{C}$ be linearly distributive categories, and $G : \mathbf{B} \longrightarrow \mathbf{C}$ be a linear functor. Then an *interpretation* for the logic LF is given by a pair of functors $[\![-]\!]_{G,S} : \mathbf{A} \longrightarrow \mathbf{B}$ and $[\![-]\!]_{G,T} : \mathbf{X} \longrightarrow \mathbf{C}$ (simply denoted $[\![-]\!]_S$ and $[\![-]\!]_T$ if the context is clear).

Theorem 1.6 (Soundness). Given an interpretation (as above), $[\![-]\!]_S$ and $[\![-]\!]_T$ lift to the free linearly distributive categories $\tilde{\mathbf{A}}, \tilde{\mathbf{X}}$, so that the following diagram commutes up to equivalence of categories:



The proof is a straightforward induction, and is left to the reader.

1.3. *Cut-elimination*

The question now arises as to whether LF satisfies cut elimination. This is, in fact, the case, though the proof requires some care in analysing the possible derivations one may have in this logic.

Theorem 1.7 (Cut-elimination). The logic LF satisfies cut-elimination, that is to say, that for every derivable sequent, there is a derivation whose only cuts are atomic.

LF also satisfies the stronger form of *categorical* cut-elimination, which would assert that derivations related to one another by cut-elimination rewrites are equal under any interpretation. We shall not make this more precise here, but it may be left to the reader familiar with the categorical context for this logic. Essentially, one must verify that each reduction step preserves equality of morphisms. For the source logic, this is exactly the traditional proof contained in Blute *et al.* (1996a). For the target logic, one must also deal with several additional rules, but they are all straightforward.

Proof. The verification that LF satisfies cut-elimination is a more-or-less straightforward structural induction. As usual, one gives an algorithm for reducing any given cut to a

cut of lesser complexity. We shall illustrate this with the one critical pair that is different from what one would meet in the logic of a linearly distributive category (such as the S entailment calculus), and which presents the characteristic new features in this induction.

Consider this cut:

$$\dfrac{\dfrac{\Gamma \vdash_S \Delta, A}{[\Gamma] \vdash_T [\Delta], F_{\otimes}(A)} \quad \dfrac{[A], \Xi \vdash_T \Upsilon}{F_{\otimes}(A), \Xi \vdash_T \Upsilon}}{[\Gamma], \Xi \vdash_T [\Delta], \Upsilon}$$

(We have seen this in Lemma 1.2, where we showed how it simulates a mixed cut. In our present context, that mixed cut would be what one might expect to reduce this cut to, but, of course, that cannot be, since such a rewrite would be an identity rewrite, hardly a successful step in a structural induction!) To see what we can reduce this cut to, we have to examine the right-hand branch of this derivation to see what sequents may lie above $[A], \Xi \vdash_T \Upsilon$. By the induction assumption, we may suppose this branch is cut-free. We claim that it is possible to rewrite the derivation using some permutations of the order in which inference rules are applied so that after these permutations we can find in the branch an S-sequent of the form $A, \Phi \vdash_S \Psi$ where the cut may be moved. The point here is that the only rules that introduce a block on the left are the $(F_{\otimes}R)$ and $(F_{\oplus}L)$ rules – apart from those, there are several rules that alter the content of such blocks, but none that introduce them. We only have to move any such 'alteration' rules up into an S sequent. Suppose, for example, that $A = A_1 \otimes A_2$ and that it is introduced by an instance of $([\otimes]L)$ somewhere in the right branch. This is the main case to consider, for it is the only case where we see an interaction between S and T sequents; for instance, by contrast, the left $\oplus$ introduction is an entirely S phenomenon, and atomic A is simple to handle. If that $([\otimes]L)$ rule follows an instance of (Mon_2L), we can easily permute these rules:

$$\dfrac{\dfrac{\cdots[\cdots\ \cdots A_1, A_2 \cdots] \cdots \vdash_T \cdots}{\cdots[\cdots][\cdots A_1, A_2 \cdots] \cdots \vdash_T \cdots}}{\cdots[\cdots][\cdots A_1 \otimes A_2 \cdots] \cdots \vdash_T \cdots} \quad \Longrightarrow \quad \dfrac{\dfrac{\cdots[\cdots\ \cdots A_1, A_2 \cdots] \cdots \vdash_T \cdots}{\cdots[\cdots\ \cdots A_1 \otimes A_2 \cdots] \cdots \vdash_T \cdots}}{\cdots[\cdots][\cdots A_1 \otimes A_2 \cdots] \cdots \vdash_T \cdots}$$

Similarly, one can permute the order of a typing rule and the tensor rule. (Note that in effect we are moving the 'bookkeeping' rules below the rules that actively generate the $[A]$, so that those rules, which must be S rules, can be pushed up into an S sequent.) Finally, we can then move this tensor introduction up to where the block is introduced, and at that point, it is easy to move it into the S calculus.

$$\dfrac{\dfrac{\cdots\ \cdots A_1, A_2 \cdots\ \cdots \vdash_S \cdots}{\cdots[\cdots A_1, A_2 \cdots] \cdots \vdash_T \cdots}}{\cdots[\cdots A_1 \otimes A_2 \cdots] \cdots \vdash_T \cdots} \quad \Longrightarrow \quad \dfrac{\dfrac{\cdots\ \cdots A_1, A_2 \cdots\ \cdots \vdash_S \cdots}{\cdots\ \cdots A_1 \otimes A_2 \cdots\ \cdots \vdash_T \cdots}}{\cdots[\cdots A_1 \otimes A_2 \cdots] \cdots \vdash_T \cdots}$$

As soon as the cut is in the S calculus, we know it can be removed. This is the main problematic critical pair in the proof; the rest we leave to the reader.

With regard to categorical cut elimination, note that the permutations we have used have no effect on the value of the derivations. □

2. The logic of linear endofunctors

In this section, we introduce logics with recursive nesting of blocks, in which the block structure reflects not only the presence of linear functors but also the presence of ordinary non-linear functors. These latter provide a very weak notion of interpretation. A particularly important example will be when we have a linear endofunctor mediating between two distinct linear structures on the same category. We shall model this by explicitly treating the identity functor as a non-linear functor that mediates the interaction between the linear structures in the reverse direction. Most interesting is the variant where the interpretation does not change the formulas at all, but just relates how the connectives interact; in categorical terms this means that not only is the linear functor an endofunctor, but the underlying functors are identities. In this case, nesting can be iterated in a very free fashion. We will show that this logic is ideal for interpreting, among other things, Abrusci and Ruet's logic NL. Before we approach these more complex cases, we shall consider a more familiar logic.

2.1. *Linear modal logic*

The simplest case where nesting of blocks can occur in the logic corresponds to when the linear functor is an endofunctor on a single category with a fixed linear structure. This gives rise to a basic linear modal logic. In modal logic it is common to write $\Box$ for $F_{\otimes}$ and $\Diamond$ for $F_{\oplus}$.

Modal (cartesian) logic (Chellas 1980; Hughes and Cresswell 1977) has many applications as the 'logic of knowledge and belief'. However, perhaps the most significant practical application was provided by Hennessy and Milner (Hennessy and Milner 1985), who showed that modal logic (suitably modified) had a complete semantics in finite processes. Excellent tutorial overviews of the use of modal logics for model-based process applications can be found in Stirling (1992) and Stirling (1994). More recently, Aldwinckle (Aldwinkle 2001) has used a proof system, very similar to the system presented in this paper, to investigate processes.

Hennessy and Milner (Hennessy and Milner 1985) actually used a generalisation of basic modal logic: they assumed a family of modalities indexed by transition labels, such as $[a]$ or $\Box_a$ for a family of necessity operators indexed by a, and $\langle a \rangle$ or $\Diamond_a$ for the corresponding possibility operators. This, of course, corresponds in our case to allowing multiple linear endofunctors: it turns out that this is a straightforward generalisation of the basic logic we now discuss.

Linear modal logic, here with one functor, may be set up in a manner very similar to the logic of a single linear functor between distinct categories. However, in this case there is only one sort of sequent and, consequently, it is possible to have nested contexts. A typical sequent may now therefore look like:

$$A_1, [A_2, [A_3, A_4]] \vdash [[B_1, B_2]].$$

The definition of the formulas and blocks is as for LF with only one atomic category **A**, and with the identification of the source and target sequents. So, formulas are generated

Table 2. *Sequent rules for* LF_{mod}

$$\frac{A \xrightarrow{f} B \text{ in } \mathbf{A}}{A \vdash B}\ (f\mathbf{A})$$

$$\frac{\mathsf{C}[\![A, B]\!] \vdash \Psi}{\mathsf{C}[\![A \otimes B]\!] \vdash \Psi}\ (\otimes L) \qquad \frac{\Phi_1 \vdash \Psi_1, A, \Psi_1' \quad \Phi_2 \vdash \Psi_2, B, \Psi_2'}{\Phi_1, \Phi_2 \vdash \Psi_1, \Psi_2, A \otimes B, \Psi_1', \Psi_2'}\ (\otimes R)$$

where either Ψ_1', Ψ_2 are empty, or Φ_1, Ψ_1' are empty, or Φ_2, Ψ_2 are empty.

$$\frac{\Phi_1, A, \Phi_1' \vdash \Psi_1 \quad \Phi_2, B, \Phi_2' \vdash \Psi_2}{\Phi_1, \Phi_2, A \oplus B, \Phi_1', \Phi_2' \vdash \Psi_1, \Psi_2}\ (\oplus L) \qquad \frac{\Phi \vdash \mathsf{C}[\![A, B]\!]}{\Phi \vdash \mathsf{C}[\![A \oplus B]\!]}\ (\oplus R)$$

where either Φ_1', Φ_2 are empty, or Φ_2, Ψ_2 are empty, or Φ_1', Ψ_1 are empty.

$$\frac{\mathsf{C}[\![\Phi, \Phi']\!] \vdash \Psi}{\mathsf{C}[\![\Phi, \top, \Phi']\!] \vdash \Psi}\ (\top L) \qquad \frac{}{\vdash \top}\ (\top R)$$

$$\frac{}{\bot \vdash}\ (\bot L) \qquad \frac{\Phi \vdash \mathsf{C}[\![\Psi, \Psi']\!]}{\Phi \vdash \mathsf{C}[\![\Psi, \bot, \Psi']\!]}\ (\bot R)$$

$$\frac{\Phi_1 \vdash \Psi_1, A, \Psi_1' \quad \Phi_2, A, \Phi_2' \vdash \Psi_2}{\Phi_2, \Phi_1 \vdash \Psi_2, \Psi_1}\ (cut)$$

where one of Φ_2, Ψ_1 is empty, and one of Φ_2', Ψ_1' is empty.

$$\frac{\mathsf{C}[\![[A]]\!] \vdash \Psi}{\mathsf{C}[\![\Box A]\!] \vdash \Psi}\ (\Box L) \qquad \frac{\Gamma \vdash \Delta, A, \Delta'}{[\Gamma] \vdash [\Delta], \Box A, [\Delta']}\ (\Box R)$$

$$\frac{\Gamma, A, \Gamma' \vdash \Delta}{[\Gamma], \Diamond A, [\Gamma'] \vdash [\Delta]}\ (\Diamond L) \qquad \frac{\Phi \vdash \mathsf{C}[\![[A]]\!]}{\Phi \vdash \mathsf{C}[\![\Diamond A]\!]}\ (\Diamond R)$$

$$\frac{\Phi, [\Gamma, \Gamma'], \Phi' \vdash \Psi}{\Phi, [\Gamma], [\Gamma'], \Phi' \vdash \Psi}\ (Mon_2 L) \qquad \frac{\Phi \vdash \Psi, [\Gamma, \Gamma'], \Psi'}{\Phi \vdash \Psi, [\Gamma], [\Gamma'], \Psi'}\ (Mon_2 R)$$

$$\frac{\Phi, [\top], \Phi' \vdash \Psi}{\Phi, \Phi' \vdash \Psi}\ (Mon_0 L) \qquad \frac{\Phi \vdash \Psi, [\bot], \Psi'}{\Phi \vdash \Psi, \Psi'}\ (Mon_0 R)$$

by the atoms and constants under the operations $\otimes, \oplus, \Box, \Diamond$, and blocks are generated by formulas and the empty block by concatenation and [ ] nesting. The sequent rules are almost the usual ones for the multiplicatives, except that the substitution of commas for their corresponding propositional structure can take place at any depth of nesting. This requires a notion of 'context' so that we can denote this arbitrary depth of nesting. Typically, we will denote a sequent containing a context by

$$\mathsf{C}[\![\Phi]\!] \vdash \Delta.$$

Here $\mathsf{C}[\![\Phi]\!]$ denotes a block in which a (sub)block Φ occurs. (We emphasise that Φ is intended as a single *block occurrence*, and does not mean all such appearances of the block Φ within the overall block C.) The definition of a (sub)block 'occurring' in a block may be given easily following the definition of block, so, for example, Φ occurs in $[\Phi]$, and Φ_1 and Φ_2 both occur in Φ_1, Φ_2, and so on (hereditarily). Then, given such a context $\mathsf{C}[\![\Phi]\!]$, by $\mathsf{C}[\![\Psi]\!]$ we mean the block that results from replacing the occurrence of Φ with the block Ψ.

The resulting logic is given in Table 2; the reader will note that it is very similar to

LF, except that, because of the context versions of the left tensor and right par rules, we do not need separate versions for when the connectives occur in a block and when they occur at the top level. These rules are sufficient to derive some of the basic entailments of most modal logics (or at least versions of such entailments that make sense in this linear, negation-free setting). Consider the examples of derivations in LF that we gave before; in LF_{mod} they now become derivations of the sequents $\Box(A \oplus B) \vdash \Box A \oplus \Diamond B$ and $\Box A \otimes \Box B \vdash \Box(A \otimes B)$, respectively:

$$\frac{A \oplus B \vdash A, B \quad \dfrac{\Box A \vdash \Box A}{[A] \vdash \Box A}}{[A \oplus B] \vdash \Box A, [B]} \qquad \frac{A, B \vdash A \otimes B \quad [A \otimes B] \vdash \Box(A \otimes B)}{\dfrac{[A, B] \vdash \Box(A \otimes B)}{[A], [B] \vdash \Box(A \otimes B)}}$$

In classical logic, the first is equivalent to the modal logic being 'normal', which is usually presented as $\Box(A \longrightarrow B) \longrightarrow (\Box A \longrightarrow \Box B)$. The second is half of the standard equivalence $\Box A \wedge \Box B \leftrightarrow \Box(A \wedge B)$. Note that in a linear setting, we do not expect the reverse half of this equivalence, since we do not have thinning.

The following rule is basic in the process calculus mentioned above (we omit reference to labelling for simplicity):

$$\frac{A_1, A_2, \cdots, A_m, B \vdash C_1, C_2, \cdots, C_n}{\Box A_1, \Box A_2, \cdots, \Box A_m, \Diamond B \vdash \Diamond C_1, \Diamond C_2, \cdots, \Diamond C_n}$$

But it is clear that this is just an instance of $(\Diamond L)$.

The logic LF_{mod} is then a good candidate for a 'basic linear modal logic'. From a philosophical perspective, it combines the basic properties of modality within a linear context. It is straightforward to add the exchange rule, linear negation, and other connectives as one wishes. (We shall indicate how to add negation to LF later in this section.)

LF_{mod} has as its semantics a (single) linear endofunctor on a linearly distributive category. It has a cut-elimination procedure that is very similar to that given in subsection 1.3. Clearly, it has the subformula property, and, consequently, derivability in this (propositional) logic is decidable, which is crucial in applications.

We shall now turn to the investigation of linear functors that mediate between two different linear structures on the same category. This will lead to a number of other examples that are already in the literature.

2.2. *The logics* LF_G *and* LF_I

We now consider logics that allow for a more complex iterated nesting. Categorical models are obtained by considering a linear functor F between two linearly distributive structures, together with a (non-linear) functor G in the opposite direction. A special case that will have significance for us will be when the two categories are the same category, but considered as linearly distributive categories *via* two tensor–par structures on that one category. In that case, the (non-linear) functor going in the opposite direction may be taken to be the identity functor. We have already seen that a single category can have many distinct linearly distributive structures, for instance the already mentioned example

of Retoré. Other examples are given in Blute *et al.* (2000) *via* the notion of *entropic Hopf algebra*: an entropic Hopf algebra induces several monoidal structures on its category of representations.

So our atomic data will be two categories $\mathbf{A}, \mathbf{X}$ as before. Logically, we still have two notions of entailment $\vdash_S$ and $\vdash_T$. But now we will also have two types of bracketing, denoted by $\{\ \}$ and $[\ ]$. Intuitively, $\{\ \}$ braces take something in the target logic and view it as part of the source logic (acting as a functor G), while the square brackets act as before, going from the source logic to the target logic (applying the linear functor F). We begin by extending the notion of *block* to this new setting.

Definition 2.1. (LF_G) Formulas for the source and target logics are defined as before, with the addition of one new source logic formation rule.

— If X is a source formula, then $G(X)$ is a source formula.

An S-block is defined by the following.

— If A is an S-formula, then A is an S-block. The empty sequence is an S-block.
— If Γ_1 and Γ_2 are S-blocks, so is Γ_1, Γ_2.
— If Ξ is a T-block, then $\{\Xi\}$ is an S-block.

T-blocks are defined as before.

— If X is a T-formula, then X is a T-block. The empty sequence is a T-block.
— If Ξ_1 and Ξ_2 are T-blocks, so is Ξ_1, Ξ_2.
— If Γ is an S-block, then $[\Gamma]$ is a T-block.

Then an *x-sequent* (for $x = S$ or T) is defined to be $\Phi \vdash_x \Psi$, where Φ and Ψ are x-blocks.

A particularly important instance of this logic is when G is taken to be the identity functor. In this case it is necessary not only to remove the functor itself but also to identify the domain and codomain categories.

Definition 2.2. (LF_I) This is defined as above, with the following alteration. We suppose only one category $\mathbf{A}$ of atoms, and we remove the source type constructor G. We define S and T blocks, and sequents, as above.

Since the two types of brackets can obviously be iterated, the logical rules must be presented as occurring within a context, similar to the linear modal logic LF_{mod}. In the present situation, we often decorate the context with a subscript (which is in fact optional because it may be inferred from the context), as illustrated by the sequent $\mathsf{C}[\![\Phi]\!]_x \vdash_y \Delta$. Here the subscript x (which must $= S$ or T) indicates what type Φ has (source or target), and the subscript $y := S$ or T is used as before to indicate which type of sequent is being considered, and so which type C is. Again, we emphasise that Φ is intended as a single block *occurrence*, and does not mean all such appearances of the block Φ within the overall block C. So, given such a context $\mathsf{C}[\![\Phi]\!]$, by $\mathsf{C}[\![\Psi]\!]$ we mean the block that results from replacing the occurrence of Φ with the block Ψ.

The rules for LF_G are presented in Table 3; there are new rules for the new type of nesting, and some old rules have been given a more general form using contexts. Note that the new form of the $([\otimes]L)$ and $([\oplus]R)$ rules actually includes not only the old rules of

Table 3. *Sequent rules for* LF_G *and* LF_I

All the rules for LF are included in LF_G;
the context rules extend (and so replace) the rules with the same names in LF.
The additional rules are as follows.
Note that x, y may represent either S or T.

$$\frac{\mathsf{C}[\![M, N]\!]_y \vdash_x \Psi}{\mathsf{C}[\![M \otimes_y N]\!]_y \vdash_x \Psi}\ ([\otimes]L) \qquad \frac{\Phi \vdash_x \mathsf{C}[\![M, N]\!]_y}{\Phi \vdash_x \mathsf{C}[\![M \oplus_y N]\!]_y}\ ([\oplus]R)$$

$$\frac{\mathsf{C}[\![\Phi, \Phi']\!]_y \vdash_x \Psi}{\mathsf{C}[\![\Phi, \top_y, \Phi']\!]_y \vdash_x \Psi}\ ([\top]L) \qquad \frac{\Phi \vdash_x \mathsf{C}[\![\Psi, \Psi']\!]_y}{\Phi \vdash_x \mathsf{C}[\![\Psi, \bot_y, \Psi']\!]_y}\ ([\bot]R)$$

$$\frac{\mathsf{C}[\![[A]]\!]_T \vdash_x \Psi}{\mathsf{C}[\![F_\otimes(A)]\!]_T \vdash_x \Psi}\ (F_\otimes L) \qquad \frac{\Phi \vdash_x \mathsf{C}[\![[A]]\!]_T}{\Phi \vdash_x \mathsf{C}[\![F_\oplus(X)]\!]_T}\ (F_\oplus R)$$

$$\frac{\mathsf{C}[\![\{X\}]\!]_S \vdash_x \Psi}{\mathsf{C}[\![G(X)]\!]_S \vdash_x \Psi}\ (\{\,\}EL) \qquad \frac{\Phi \vdash_x \mathsf{C}[\![\{X\}]\!]_S}{\Phi \vdash_x \mathsf{C}[\![G(X)]\!]_S}\ (\{\,\}ER)$$

$$\frac{\Xi \vdash_T \Upsilon}{\{\Xi\} \vdash_S \{\Upsilon\}}\ (\{\,\}I)$$

For the sequent calculus for LF_I,
remove instances of G above (so $G(X)$ becomes X),
and add the following rule.

$$\frac{\{X\} \vdash_S \{Y\}}{X \vdash_T Y}\ (\{\,\}E2)$$

those names, but also the ordinary linear logic rules $(\otimes L)$ and $(\oplus R)$. Table 3 also includes a rule for the system LF_I; this system corresponds to the case where the non-linear functor G is, in fact, the identity functor and so F is an endofunctor, the category $\mathbf{A}$ having both tensor–par structures. The effect of these systems is:

(a) in the case of LF_G, we construct the free linearly distributive categories $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{X}}$ as before, a linear functor F as before, and an ordinary functor $G\colon \tilde{\mathbf{A}} \longrightarrow \tilde{\mathbf{X}}$, and

(b) in the case of LF_I, we construct the free category $\tilde{\mathbf{A}}$ with two linearly distributive structures on it, using different tensor and par combinators (and different units), as well as a linear endofunctor F on $\tilde{\mathbf{A}}$.

The effect of the additional rule in LF_I is that as a derived rule we can infer a bijection between sequents $M \vdash_S N$ and $M \vdash_T N$ (for single formulas M, N). This bijection does not extend to arbitrary blocks, since there are two distinct tensor–par structures for S and T.

As in the situation with LF, it is easy to note that the rules $([\otimes])$ and $([\oplus])R$ are in fact bijections, as are the unit rules $([\top]L)$ and $([\bot]R)$. In fact, we shall soon see that all the rules in Table 3 are bijective. There are also derived mixed cut rules, such as

$$\frac{\Gamma' \vdash_T \Delta', X \quad \{X, \Gamma\} \vdash_S \{\Delta\}}{\Gamma', \Gamma \vdash_T \Delta', \Delta}$$

and all the planar variants. In fact, one can also derive a general 'context cut' rule.

Lemma 2.3. The following context cut rule is derivable in LF_G:

$$\frac{\Phi \vdash_x M \quad \mathsf{C}[\![M]\!]_x \vdash_y \Psi}{\mathsf{C}[\![\Phi]\!]_x \vdash_y \Psi}$$

where M is a single formula.

Proof. The proof is done by induction on the structure of the context in which Φ, M appear. Clearly, the rule is derivable for basic contexts; we shall illustrate the induction by supposing it true for a context, say $\mathsf{C}'[\![A]\!]_S$, and show that it is true for a context constructed from C', say $\mathsf{C}[\![A]\!] = \mathsf{C}'[\![\{\Xi_1, [\Gamma_1, A, \Gamma_2], \Xi_2\}]\!]$. The key here is to note that the commas and nesting brackets are representable *via* derivable sequents such as $\{X\} \vdash_S G(X)$ and derived rules such as

$$\frac{\Gamma \vdash_S A}{\{\Xi_1, [\Gamma_1, \Gamma, \Gamma_2], \Xi_2\} \vdash_S G(\Xi_1 \otimes F_{\otimes}(\Gamma_1 \otimes A \otimes \Gamma_2) \otimes \Xi_2)}$$

So, continuing the induction example we started, we can do the context cut on $\mathsf{C}[\![A]\!] = \mathsf{C}'[\![\{\Xi_1, [\Gamma_1, A, \Gamma_2], \Xi_2\}]\!]$, given that we know we can cut contexts of the form $\mathsf{C}'[\![A]\!]$, as follows:

$$\frac{\dfrac{\Gamma \vdash A}{\{\Xi_1, [\Gamma_1, \Gamma, \Gamma_2], \Xi_2\} \vdash G(\Xi_1 \otimes F_{\otimes}(\Gamma_1 \otimes A \otimes \Gamma_2) \otimes \Xi_2)} \quad \dfrac{\dfrac{\mathsf{C}'[\![\{\Xi_1, [\Gamma_1, A, \Gamma_2], \Xi_2\}]\!] \vdash \Psi}{\mathsf{C}'[\![G(\Xi_1, [\Gamma_1, A, \Gamma_2], \Xi_2)]\!] \vdash \Psi}}{\mathsf{C}'[\![G(\Xi_1 \otimes F_{\otimes}(\Gamma_1 \otimes A \otimes \Gamma_2) \otimes \Xi_2)]\!] \vdash \Psi}}{\mathsf{C}'[\![\{\Xi_1, [\Gamma_1, \Gamma, \Gamma_2], \Xi_2\}]\!] \vdash \Psi}$$

The other cases of the induction are either similar, or simpler. □

Note that the entropy and typing rules only hold for the square brackets, as before, and not for the new { } braces, because only the 'forward' direction corresponds to a linear functor. In addition, LF_I has the following two-way derived rules, called *phase change* in the Abrusci–Ruet terminology:

$$\frac{\{\Xi\} \vdash_S \{\Upsilon\}}{\Xi \vdash_T \Upsilon} \qquad \frac{\mathsf{C}[\![\{X\}]\!] \vdash_x \Psi}{\mathsf{C}[\![X]\!] \vdash_x \Psi} \qquad \frac{\Phi \vdash_x \mathsf{C}[\![\{X\}]\!]}{\Phi \vdash_x \mathsf{C}[\![X]\!]}$$

In LF_G the context rules above are also two-way, replacing X with $G(X)$. Also, the context versions of the typing rules are two-way in LF_G. Consider the 'down' direction of the first of these rules:

$$\frac{\{\Xi\} \vdash_S \{\Upsilon\}}{\Xi \vdash_T \Upsilon}$$

(the other direction is already a rule of LF_G). We can represent Ξ, Υ as single T-formulas, so this rule is a consequence of ({ }$E2$). The only subtle point is that the representation of Ξ, Υ in both sequents is in terms of the T connectives, even though one of the sequents, the premise, is an S-sequent, since Ξ, Υ appear in a { } block. We shall leave the proof of the others to the reader, pointing out that in each case one direction is already in the rules, and the other follows from context cut.

In addition, we have made sufficient comment about categorical semantics for the reader

to realize that we have constructed a universal solution to the model we described in the introductory remarks, so we have notions of valuation and soundness (and completeness) that express that fact, as we outlined for LF.

2.2.1. *Cut elimination* Cut elimination for LF_G and LF_I are proved using a similar technique to LF. Again, we can permute rules so as to 'push up' an active formula so that the cut can be moved up as necessary. For example, suppose we have a cut with $G(X)$ as active formula:

$$\frac{\Gamma \vdash_S G(X), \Delta \quad \dfrac{\Gamma', \{X\} \vdash_S \Delta'}{\Gamma', G(X) \vdash_S \Delta'}}{\Gamma, \Gamma' \vdash_S \Delta, \Delta'}$$

We want to move the cut up the right branch of this derivation, which means we want to move it up into a T-sequent. Considering what rules we have for introducing $G(X)$ and $\{\ \}$ into the derivation, assuming the derivation is cut-free above the displayed cut, and allowing for some permutation of deduction rules, we must be able to find instances of $(\{\ \}EL)$ (on the left) and so $(\{\ \}I)$ (on the left and on the right) so that we can move the cut up to a T-cut higher up the tree.

Hence we can prove the following cut elimination theorems.

Theorem 2.4 (Cut-elimination). The logics LF_G and LF_I satisfy cut-elimination, that is to say that for every derivable sequent, there is a derivation whose only cuts are atomic.

2.2.2. *The logic* LF_{can} Finally, we introduce a variant of LF_I that allows us to link up with another well-known logic; this corresponds semantically to requiring that the linear functor F be an endofunctor, but, additionally, that the underlying functors $F_\otimes, F_\oplus$ are both the identity. The resulting logic is the logic LF_{can}, which is LF_I augmented by the additional two-way rules of *cancellation*, as follows:

$$\frac{\mathsf{C}[\![\, \{[\Phi]\} \,]\!]_x \vdash_y \Psi}{\mathsf{C}[\![\, \Phi \,]\!]_x \vdash_y \Psi} \qquad \frac{\mathsf{C}[\![\, [\{\Phi\}] \,]\!]_x \vdash_y \Psi}{\mathsf{C}[\![\, \Phi \,]\!]_x \vdash_y \Psi}$$

and the duals on the right. These may be derived from simpler cases, and the formal new rules for LF_{can} are all given in Table 4. These rules force the components of the linear functor to be identities. It is also of interest to note that in this case, if the target category $(\mathbf{A}, \otimes_T, \oplus_T)$ is symmetric, the logic of the source category $(\mathbf{A}, \otimes_S, \oplus_S)$ is forced to be cyclic (which in a sense is 'why' the noncommutative operators in NL are cyclic and not arbitrarily noncommutative). The other implication is also true: if the source is symmetric, the target must be cyclic.

To see that the context versions are derivable, we use the usual procedure of cutting with derivable sequents, in this case of the form $\{[A]\} \vdash_S A$, and the evident three variants. Similarly, the cut elimination proof extends to this logic. The only new cuts we need to worry about are those where the active formula is the result of an application of a 'double nesting elimination' rule (which eliminates $\{[\ ]\}$ or $[\{\ \}]$); again, we 'push' the cut up the branch until we get an unnested occurrence of the formula involved.

Table 4. *Sequent rules for* LF_{can}

All the rules for LF, LF_I are included in LF_{can}; the additional cancellation rules are as follows.

$$\frac{\{[\Gamma]\} \vdash_S \Delta}{\Gamma \vdash_S \Delta}\ (\{[\,]\}EL) \qquad \frac{\Gamma \vdash_S \{[\Delta]\}}{\Gamma \vdash_S \Delta}\ (\{[\,]\}ER)$$

$$\frac{\Gamma \vdash_S \Delta}{\{[\Gamma]\} \vdash_S \Delta}\ (\{[\,]\}IL) \qquad \frac{\Gamma \vdash_S \Delta}{\Gamma \vdash_S \{[\Delta]\}}\ (\{[\,]\}IR)$$

$$\frac{[\{\Xi\}] \vdash_T \Upsilon}{\Xi \vdash_T \Upsilon}\ ([\{\,\}]EL) \qquad \frac{[\{\Xi\}] \vdash_T \Upsilon}{\Xi \vdash_T \Upsilon}\ ([\{\,\}]EL)$$

$$\frac{\Xi \vdash_T \Upsilon}{[\{\Xi\}] \vdash_T \Upsilon}\ ([\{\,\}]IL) \qquad \frac{\Xi \vdash_T \Upsilon}{[\{\Xi\}] \vdash_T \Upsilon}\ ([\{\,\}]IL)$$

Why do these new rules force F to be the identity? This essentially follows from the observation that the linear functor F represents the [ ] nesting, and the identity functor (in LF_I – in LF_G this role is played by the functor G) represents the { } nesting, so the cancellation laws make F be the identity. To understand our other comment, that if the target is symmetric, then the source must be cyclic, we must see how to add negation to this setting. Once we have done that, we will then also see that the resulting system LF_{can} with negation is equivalent to NL, in the sense that there are interpretations from each to the other, preserving validity of sequents.

2.2.3. *Negation* We have seen how, in the context of linearly distributive categories, we can add a negation operator together with natural rules that make it involutive. If the tensor–par structure is commutative, one negation operator suffices, but in the general noncommutative context, one would add both left and right negation operators. If they should coincide, that would make the logic 'cyclic'. For the present purposes, we shall present this in terms of two-way rules, which express the adjunctions that negation represents. We suppose that there are four negation operators, a left and a right negation for each of S, T. We shall subscript them in the evident way, as we did with the tensor, par, and their units:

$$\frac{\Phi, A \vdash_x \Psi}{\Phi \vdash_x \Psi, {}^{\perp_x}A} \qquad \frac{{}^{\perp_x}A, \Phi \vdash_x \Psi}{\Phi \vdash_x A, \Psi} \qquad \frac{A, \Phi \vdash_x \Psi}{\Phi \vdash_x A^{\perp_x}, \Psi} \qquad \frac{\Phi, A^{\perp_x} \vdash_x \Psi}{\Phi \vdash_x \Psi, A}$$

These rules may be added to any of the logics considered above, but perhaps it is simplest to regard them as an addition to LF, since that logic is at the base of our systems. We shall use $\mathsf{LF}^{\neg}$ to denote the system LF with such a family of negation operators. This then induces the system $\mathsf{LF}^{\neg}_{can}$, which is LF_{can} plus negation. However, there is a more efficient presentation of $\mathsf{LF}^{\neg}_{can}$, which reduces the number of negation operators to two. For, consider this semantically: we know for a linear functor F defined on $*$-autonomous categories $\mathbf{A} \longrightarrow \mathbf{X}$ that $F_{\otimes}({}^{\perp_S}A) \cong {}^{\perp_T}F_{\oplus}(A)$ (and similarly for the right negation operators) (Cockett and Seely 1999). If F is given by the identity functor (that is, both $F_{\otimes}$ and $F_{\oplus}$ are the identity) on a $*$-autonomous category $\mathbf{A}$, then this says that ${}^{\perp_S}A \cong {}^{\perp_T}A$. And, similarly, we have $A^{\perp_S} \cong A^{\perp_T}$. (It is a simple exercise to verify that these equivalences are derivable

in the logic $\mathsf{LF}^{\neg}_{can}$ just using the two-way rules above together with the fact that in $\mathsf{LF}^{\neg}_{can}$, in fact in LF_I, there is a bijection between sequents $M \vdash_S N$ and $M \vdash_T N$ between single formulas.) This suggests that a more efficient formulation of $\mathsf{LF}^{\neg}_{can}$ would be to have just one left negation operator ${}^{\perp}A$ and one right negation operator $A^{\perp}$, satisfying the adjunction (two-way) rule above, for both S and T sequents.

With this presentation of $\mathsf{LF}^{\neg}_{can}$, suppose that one of the logics, S or T, is in fact commutative, meaning that the exchange law holds. Categorically, this would mean that the tensor and par are commutative. Then in that logic the two negation operators would coincide, and that would then immediately mean that they coincide in the other logic. It would not be necessary for the tensor and par to be commutative in that logic, but it would mean that that logic is 'cyclic' in the sense of Yetter (Yetter 1990).

Categorically, there is a tiny bit more happening here, and this is discussed fully in Cockett *et al.* (2000). Essentially, if we have an '$\mathsf{LF}^{\neg}_{can}$' setting (F between $*$-autonomous categories being given by identity), then, if one of the two linearly distributive structures is symmetric, the other must be cyclic, which in addition to having the two negation operators coincide, means that a certain coherence condition due to Rosenthal (Rosenthal 1992) must hold. This extra coherence condition turns out to be trivial in this setting. As this is not the focus of the present paper, we shall not develop these details further.

We conclude this discussion with a comment on how nested blocks may be negated. In $\mathsf{LF}^{\neg}$ we may derive the following correspondence:

$$\frac{[\Gamma], \Xi \vdash_T \Upsilon}{\Xi \vdash_T [\Gamma^{op\perp}], \Upsilon}$$

(as well as three others similar to the ones for a single formula). By Γ^{op} we mean to reverse the order of the formulas in Γ; by $\Gamma^{\perp}$ we mean that each formula of Γ is negated. This corresponds to the isomorphism $F_{\otimes}(A \otimes B)^{\perp} \cong F_{\oplus}(B^{\perp} \oplus A^{\perp})$. In LF_I, once we have G the identity, we can also 'negate' $\{\ \}$ blocks in a similar manner.

3. Mixed logics are functor logics

The key new idea contained in these various notions of logic associated to a functor is the idea of a logic that incorporates two distinct logical systems, and allows them to interact. While this idea has occurred in several places previously, the current work allows for a unified treatment. The idea of interacting logics is prevalent in the examples we now consider. We have already noted that simple modal logics fit well into the framework given by LF, and since Retoré's 'next' operator on coherence spaces has a canonical interpretation as a linear functor (with underlying functor the identity), our logic LF_{can} provides a logical syntax for its analysis. We now consider two other examples.

3.1. *Abrusci and Ruet's noncommutative logic* NL

This example was one of the main motivations for developing the logic LF, which provides a context for a general semantic analysis of NL. The key principle behind the logic NL is that it combines a commutative and noncommutative component. NL

as originally presented contained negation, and so semantically corresponds to a $*$-autonomous category. Recall that in the $*$-autonomous context, linear functors and transformations correspond to monoidal functors and transformations, the comonoidal components being implicitly induced by duality. So, a model of NL could be viewed as a category equipped with a symmetric tensor making it into a (symmetric) $*$-autonomous category, and a second monoidal structure making it into a cyclic $*$-autonomous category (Yetter 1990; Blute and Scott 1998). Then the two monoidal structures are related by a monoidal transformation called the entropy map. This is summarised in the following definition, which was presented in Blute *et al.* (2000) and is an example of the approach of the present paper.

Definition 3.1. An *entropic category* is a category C equipped with:

1 two functors $\circ, \otimes$ and an object $\top$ in C such that $(\mathsf{C}, \circ, \top)$ is a monoidal category and $(\mathsf{C}, \otimes, \top)$ is a symmetric monoidal category (with same unit);
2 a monoidal natural transformation $\epsilon : \otimes \longrightarrow \circ$, called *entropy* ;
3 an equivalence $(-)^* : \mathsf{C} \to \mathsf{C}^{op}$ such that there are natural isomorphisms

$$\begin{aligned} & Hom(A, B^*) \cong Hom(\top, (A \circ B)^*) \\ \text{and} \quad & Hom(A, B^*) \cong Hom(\top, (A \otimes B)^*) \end{aligned}$$

satisfying certain coherence conditions.

3.1.1. *Bigroups* To illustrate that there is a wealth of entropic categories, we summarise one class of examples described in Blute *et al.* (2000).

Definition 3.2 (Bigroup). A bigroup consists of a set X with the following additional structure: $X = (X, \circ, *, S_1, S_2, 1)$ where $\circ$ and $*$ are multiplications on X, S_1 and S_2 are endomorphisms of X and 1 is an element of X such that $(X, \circ, S_1, 1)$ and $(X, *, S_2, 1)$ are both groups (S_i acting as inverse). A bigroup is *strong* if $S_1 = S_2$.

The point about bigroups is that if X is a strong finite bigroup, then the category $\mathsf{RTComod}(X)$ of reflexive topological finite dimensional comodules is entropic, and even if X is not strong, there is a Chu construction to make an entropic category. Methods of constructing bigroups are described in Blute *et al.* (2000). In addition, the authors show that the definition is sound, that is, that proofs in NL can be interpreted canonically as morphisms in an entropic category, and, furthermore, the interpretation is sound under cut-elimination.

3.1.2. *Relation between* NL *and* LF_{can} The present work provides a closely related formulation of the semantics of NL, which is somewhat more general, since negation need not be part of the structure. Specifically, a model can be defined as a category **C** together with two $*$-autonomous structures, one symmetric (and so the other is cyclic), and appropriate additional structure to make the identity a linear functor between the two $*$-autonomous structures. Thus NL is an instance of our logic LF_{can}. We could say that LF_{can} is the 'two-sided tensor–par negation-free fragment' of NL. We should point out that one of the advantages of the present approach is that it shows how one can modularise the approach

taken by Abrusci and Ruet, allowing one to adjust the features of the logic if one wishes. But it is simple enough to get a logical syntax equivalent to NL in the spirit of LF_{can}.

We shall make this statement somewhat more precise now by showing how one can interpret each logic into the other. To make this possible, however, we have to modify LF_{can} a bit more. NL has only one unit for the two tensors and one for the two pars: $\top_S = \top_T$ and $\bot_S = \bot_T$. This is needed because Abrusci and Ruet insist on one-sided sequents, in the linear logic tradition. For example, if one considers the two-way rule (which is part of NL)

$$\frac{\vdash A \ ; B}{\vdash A, B}$$

where they use a comma to indicate the commutative binder and a semicolon for the noncommutative binder, it is clear that since they have only one type of sequent, they will need to have the same place holder for the empty left-hand side of the sequent. In other words, they need $\top_S = \top_T$ in our terminology. So, we shall also assume for this interpretation that the units in LF_{can} coincide as above. Furthermore, we ought also to suppose that the target tensor and par are commutative. We could assume negation (cyclic in the source) if we wanted, using (a slight modification of) $\mathsf{LF}^{\neg}_{can}$, but, instead, we shall content ourselves with translating one-sided sequents.

[LF_{can} to NL] We can inductively define a translation $\Phi \mapsto \overline{\Phi}$ for blocks Φ of LF_{can}. The translation depends on whether the block is of S or T type; we shall write $\Phi : x$ to indicate that Φ is an x-block ($x = S$ or T). For single formulas, $\overline{M} = M$ in either case. For the units, $\overline{\{\,\}:S} = \bot$ and $\overline{[\,]:T} = \bot$ (if we were translating occurrences of these on the left, we would use $\top$ instead). The rest of the induction is as follows:

— $\overline{\Gamma_1, \Gamma_2 : S} = (\overline{\Gamma_1 : S} \ ; \overline{\Gamma_2 : S})$
— $\overline{\{\Xi\} : S} = \overline{\Xi : T}$
— $\overline{\Xi_1, \Xi_2 : T} = (\overline{\Xi_1 : T} \, , \overline{\Xi_2 : T})$
— $\overline{[\Gamma] : T} = \overline{\Gamma : S}$.

[NL to LF_{can}] For the reverse direction we shall have two translations, $\Phi \mapsto \overline{\Phi}^S$ and $\Phi \mapsto \overline{\Phi}^T$, into the source and target logics. Recall that the semicolon acts as the binder in the 'source' (*viz.* the noncommutative) part of NL and the comma in the 'target' (or commutative) part. We translate ordinary formulas as themselves in both cases: $\overline{M}^S = \overline{M}^T = M$. The other steps of the induction are as follows:

— $\overline{\Phi \ ; \Psi}^T = [\overline{\Phi}^S, \overline{\Psi}^S]$
— $\overline{\Phi \ ; \Psi}^S = \overline{\Phi}^S, \overline{\Psi}^S$
— $\overline{\Phi, \Psi}^T = \overline{\Phi}^T, \overline{\Psi}^T$
— $\overline{\Phi, \Psi}^S = \{\overline{\Phi}^T, \overline{\Psi}^T\}$.

For example, the NL sequent $\vdash (A \, , B) \ ; (C \ ; D)$ is translated as follows:

$$\begin{aligned} \overline{(A \, , B) \ ; (C \ ; D)}^S &= \overline{A \, , B}^S, \overline{C \ ; D}^S \\ &= \{\overline{A}^T, \overline{B}^T\}, \overline{C}^S, \overline{D}^S \\ &= \{A, B\}, C, D \end{aligned}$$

and

$$\begin{aligned}\overline{(A\,,\,B)\;;(C\;;D)}^{T} &= [\overline{A\,,\,B}^{S}, \overline{C\;;D}^{S}]\\ &= [\{\overline{A}^{T}, \overline{B}^{T}\}, \overline{C}^{S}, \overline{D}^{S}]\\ &= [\{A, B\}, C, D].\end{aligned}$$

We shall leave the reasonably straightforward proof that these translations are sound to the reader. As an example, we give the interpretation of an instance of 'entropy':

$$\frac{\vdash (A\,,\,B)\;;(C\;;D)}{\vdash (A\,,\,B)\,,\,(C\;;D)}$$

becomes, when interpreted as a T-sequent of LF_{can},

$$\frac{\vdash_T [\{A\,,\,B\}\,,\,C\,,\,D]}{\vdash_T A\,,\,B\,,\,[C\,,\,D]}$$

which may be proved as follows:

$$\frac{\dfrac{\vdash_T [\{A\,,\,B\}\,,\,C\,,\,D]}{\vdash_T [\{A\,,\,B\}]\,,\,[C\,,\,D]}}{\vdash_T A\,,\,B\,,\,[C\,,\,D]}$$

3.2. *Affine bunched logic*

One point of the present paper is that we have developed an approach for logics with related connectives with a sound ideological doctrine behind it, namely that of linear functors. That doctrine has suggested a notational approach based on nested blocks, which is somewhat contrasted to the main alternative, 'bunching'. We think the Abrusci–Ruet logic is an excellent example of this 'bunching', but there are others: one, in particular, due to O'Hearn and Pym, uses this name explicitly. Their bunched logic (O'Hearn and Pym 1999; O'Hearn 2000) may be used to give a logical analysis of Reynolds' notion of *syntactic control of interference* (Reynolds 1978). For example, one of the key notions there is the distinction between *active* and *passive* types. Passive types are subject to more structural rules. In particular, one has contraction with respect to such types, as it is contraction that allows for the appropriate model of sharing. So, a proper logical framework would require two interacting logical structures, one being (multiplicative) linear, and the other being intuitionistic. This is precisely what bunched logic does. One models the two logical structures by a monoidal closed and a (possibly different) cartesian closed structure on the same underlying category. Analogously, one must have two types of binding. This leads to a variant of the λ-calculus, which the authors call the $\lambda\alpha$-calculus.

This situation seems close to the work presented here, but there are some significant differences. In the most basic form of bunched logic, the two logical structures (imposed on the same category) exist independently of one another. There is no interaction between the two logics. However, O'Hearn (O'Hearn 2000) introduces a variant, which he calls *affine bunched logic*. This has the additional stipulation that the two units coincide. It is straightforward to see that as a result one obtains a natural transformation of the form

$A \otimes B \longrightarrow A \times B$, which makes the identity a monoidal functor, and thus introduces the sort of relationship between the two structures that we have considered here. However, the O'Hearn–Pym bunched logic is, essentially, not a fragment of linear logic, as the authors point out, and so cannot be presented as a variant of any of the logics presented here without considerable distortion to the other operators, but clearly it would fit into a somewhat more general 'functor logic' framework, illustrating that this context has further generality than the examples in this paper.

4. Conclusion

One of the main points to notice about the preceding is that we have presented a uniform context for many different phenomena. The block presentation (as opposed to the bunch presentation) of logics with mixed substructures provides considerable flexibility within a coherent framework. There are many instances of the logics presented in this paper in other contexts. For example, both the exponential and additive connectives of linear logic provide examples of linear functors on a $*$-autonomous category. Thus, our logics LF and LF_I provide an alternate sequent calculus presentation for these fragments, and the functor nets of Cockett and Seely (1999) provide a net-theoretic syntax. In the exponential case, these will be a variant of the familiar exponential boxes introduced by Danos and Regnier (Danos 1990) and studied from a categorical perspective in Blute *et al.* (1996b) and Cockett and Seely (1999).

We have also noted that our logic LF_{can} provides an alternative sequent calculus for the analysis of Retoré's noncommutative connective on coherence spaces. It should be of interest to compare our sequent calculus with that introduced by Retoré. In his work, there is a notion of *ordered sequent*, that is, there is a partial order on formulas in the sequent. The relationship to our notion of nesting should be worth examining.

We have provided a general context for the semantic analysis of NL, which includes the approach of Blute *et al.* (2000) *via* entropic categories. That work continues with Blute *et al.* (2000) and its sequels.

Finally, there are close connections between this linear functor logic and model-based process applications *via* the modal logics of Hennessy and Milner, which suggests applications of this approach in the study of finite processes (Aldwinkle 2001).

References

Abrusci, V. and Ruet, P. (2000) Noncommutative logic I: the multiplicative fragment. *Annals of Pure and Applied Logic* **101** 29–64.

Aldwinckle, J. (2001) Doctoral dissertation (in preparation), University of Calgary, Department of Computer Science.

Atiyah, M. (1990) *The geometry and physics of knots*, Accademmia Nazionale Dei Lincei.

Barr, M. (1979) $*$-Autonomous Categories. *Springer-Verlag Lecture Notes in Mathematics* **752**.

Barr, M. (1995) Nonsymmetric $*$-autonomous categories. *Theoretical Computer Science* **139** 115–130.

Blute, R. F. (1993) Linear logic, coherence and dinaturality. *Theoretical Computer Science* **115** 3–41.

Blute, R. F., Cockett, J. R. B., Seely, R. A. G. and Trimble, T. H. (1996a) Natural deduction and coherence for weakly distributive categories. *Journal of Pure and Applied Algebra* **113** 229–296.

Blute, R. F., Cockett, J. R. B. and Seely, R. A. G. (1996b) ! and ?: Storage as tensorial strength. *Mathematical Structures in Computer Science* **6** 313–351.

Blute, R., Lamarche, F. and Ruet, P. (2000) Hopf algebras and models of noncommutative logic (to appear in *Theory and Applications of Categories*).

Blute, R. and Scott, P. (1998) The shuffle Hopf algebra and noncommutative full completeness. *Journal of Symbolic Logic* **63** 1413–1436.

Chellas, B. (1980) *Modal Logic: An Introduction*, Cambridge University Press.

Cockett, J. R. B., Koslowski, J. and Seely, R. A. G. (2000) Introduction to linear bicategories. *Mathematical Structures in Computer Science* **10** 165–203.

Cockett, J. R. B. and Seely, R. A. G. (1997a) Weakly distributive categories. *Journal of Pure and Applied Algebra* **114** 133–173. (See `http://www.math.mcgill.ca/rags` for an updated version.)

Cockett, J. R. B. and Seely, R. A. G. (1997b) Proof theory for full intuitionistic linear logic, bilinear logic, and MIX categories. *Theory and Applications of Categories* **3** 85–131.

Cockett, J. R. B. and Seely, R. A. G. (1999) Linearly distributive functors. *Journal of Pure and Applied Algebra* **143** 155–203.

Danos, V. (1990) *La logique linéaire appliquée à l'étude de divers processus de normalization et principalement du λ-calcul*, Ph. D. dissertation.

Girard, J.-Y. (1987) Linear logic. *Theoretical Computer Science* **50** 1–102.

Hennessy, M. and Milner, R. (1985) Algebraic Laws for Nondeterminism and Concurrency. *Journal of the ACM* **32** 137–161.

Hughes, G. E. and Cresswell, M. E. (1977) *An Introduction to Modal Logic*, Methuen and Co Ltd.

Maielli, R. and Ruet, P. (2000) Noncommutative logic III: focusing proofs (preprint).

Majid, S. (1995) *Foundations of quantum group theory*, Cambridge University Press.

O'Hearn, P. W. (2000) On bunched typing (preprint).

O'Hearn, P. W. and Pym, D. J. (1999) The logic of bunched implications. *Bulletin of Symbolic Logic* **5** 215–244.

Quinn, F. (1995) Lectures on Axiomatic Topological Quantum Field Theory. In: Freed, D. S. and Uhlenbeck, K. K. (eds.) *Geometry and Quantum Field Theory*, American Mathematical Society Institute for Advanced Study/Park City Mathematics Series **1** 325–453.

Retoré, C. (1997) Pomset logic: a noncommutative extension of classical linear logic. In: Proceedings TLCA'97. *Springer-Verlag Lecture Notes in Computer Science* **1210** 300–318.

Reynolds, J. C. (1978) Syntactic control of interference. In: O'Hearn, P. and Tennent, R. (eds.) *Algol-like languages*, Birkhauser.

Rosenthal, K. I. (1992) Girard quantaloids. *Mathematical Structures in Computer Science* **2** (1) 93–108.

Ruet, P. (2000) Noncommutative logic II, sequent calculus and phase semantics. *Mathematical Structures in Computer Science* **10** 277–312.

Schneck, R. R. (1998) Natural deduction and coherence for non-symmetric linearly distributive categories. *Theory and Applications of Categories* **6** 105–146.

Seely, R. A. G. (1989) Linear logic, *-autonomous categories and cofree coalgebras. In: Gray, J. and Scedrov, A. (eds.) Categories in Computer Science and Logic. *Contemporary Mathematics* **92** 371–382.

Stirling, C. (1992) Modal and Temporal Logics. In: Abramsky, S., Gabbay, D. M. and Maibaum, T. S. E. (eds.) *Handbook of Logic in Computer Science* **2**, Cambridge University Press 477–563.

Stirling, C. (1994) Logics for Concurrency. In: Moller, F. and Birtwistle, G. (eds.) Modal and Temporal Logics for Processes. *Springer-Verlag Lecture Notes in Computer Science* **1043** 139–237.

Yetter, D. (1990) Quantales and (noncommutative) linear logic. *Journal of Symbolic Logic* **55** 41–64.