Proof Theory of the Cut Rule

J.R.B. Cockett R.A.G. Seely

1 Introduction

The cut rule is a very basic component of any sequent-style presentation of a logic. This essay starts by describing the categorical proof theory of the cut rule in a calculus which allows sequents to have many formulas on the left but only one on the right of the turnstile. We shall assume a minimum of structural rules and connectives: in fact, we shall start with none. We will then show how logical features can be added to this proof theoretic substrate in a modular fashion. The categorical semantics of the proof theory of this modest starting point, assuming just the cut rule, lies in multicategories: we shall refer to the resulting logic as **multi-logic**[1] to emphasize this connection.

The list of formulas to the left of the turnstile are separated by commas: this comma may be "represented" by a logical connective called "tensor", written $\otimes$: we may regard this connective as a primitive conjunction which lacks the usual structural rules of weakening and contraction. When this connective is present it is usual to also "represent" the empty list of formulas with a constant called the "tensor unit", written $\top$, which may be regard as a primitive "true". Of course, truth and falsity in these logics is not the central issue: rather the main interest is how proofs in these logics behave. When these "representing" connectives are assumed to be present we call the result $\otimes$-multi-logic. Significantly, the categorical semantics of the proof theory of a $\otimes$-multi-logic lies in the doctrine of monoidal categories.

Our next step is to consider logics whose sequents have many formulas both on the left and right of the turnstile. Again we assume no structural rules and no connectives, and start with just the cut rule, adapted, however, to this two-sided setting. The resulting logic then has its categorical semantics in polycategories[2] and consequently we shall refer to it as **poly-logic**.

Again we may add connectives to represent the commas. This time, however, we need two different connectives: one for the commas on the left, given by "tensor", and one for the commas on the right, given by a "par", written $\oplus$. We may regard the latter as a primitive disjunction which lacks the usual structural rules. Both the tensor and par connectives have units: the unit for the par, written $\bot$, may be regarded as a primitive "false". The categorical semantics of the proof theory of this very minimal logic with connectives then lies in the doctrine of linearly distributive categories.

[1] The "multi-" prefix indicates that one can have a list of formulas to the left of the turnstile but only one formula to the right.

[2] The "poly-" prefix indicates that one can have a list of formulas both to left and to the right the turnstile.

In the presence of negation, this logic is precisely the multiplicative fragment of linear logic[3]. When a formula is negated, it moves onto the other side of the turnstile: thus, in the presence of negation, it is possible to reduce a two-sided sequent to a one-sided sequent. This meant that, in the development of linear logic, the behaviour of the $\otimes\oplus$-poly-logic could be—and was—avoided. In particular, this meant that, the categorical semantics of two-sided proof systems was also avoided. Thus, the fundamental importance to these systems of the natural transformation

$$\delta\colon A \otimes (B \oplus C) \longrightarrow (A \otimes B) \oplus C$$

called a linear distribution, was overlooked.

A linear distribution may be viewed as a "tensorial strength" in which the object A is pushed into the structure $B \oplus C$. Dually, it may be viewed as a "tensorial costrength" in which the object $A \otimes B$ is the structure which remains after pulling out C. These linear notions of strength pervade the features of poly-logics, and thus in particular of linear logic, and provide a unifying structure for them.

It is important to note that, even in this very basic $\otimes\oplus$-poly-logic, the behaviour of the—so called—"multiplicative" units (*i.e.* the unit of the tensor, $\top$, and the unit of the par, $\bot$) is very subtle. Indeed, it is the behaviour of the units at this very basic level that makes deciding the equality of proofs difficult; exactly how difficult was an open problem until recently: for multiplicative linear logic with units, deciding equality is PSPACE complete [HH14], though without units the problem is linear.

The full proof theory of linear logic can be built in a modular fashion from the basic semantics of linearly distributive categories. Negation, that is the requirement that every object have a complement, is fundamental to linear logic. For a linearly distributive category, having a complement is a property rather than extra structure: this means in a linearly distributive category an object either has a negation or it does not. It is, furthermore, possible to "complete" a linearly distributive category by formally adding negations. A linearly distributive category in which every object has a complement is precisely a $*$-autonomous category and $*$-autonomous categories provide precisely the categorical semantics for the—so called—"multiplicative fragment" of linear logic.

In the posetal case, the fact that complements are a property is exactly the observation that in a distributive lattice either an element has a complement or it does not. Of course, as is well-known, a distributive lattice in which every object has a complement is a Boolean algebra. It is worth remarking that the fact that a distributive lattice can always be embedded in a Boolean algebra has not made these lattices a lesser area of study. Quite the converse is true: the theory of distributive lattices has hugely enriched the development and understanding of ordered structures. The reason and motivation for studying more general structures, such as linearly distributive categories, in the context of linear logic is exactly analogous: it enriches and broadens our understanding of the "linear" world.

To arrive at the full structure of linear logic, after negation we require the presence of the

[3]Readers should note that we use a notation different from Girard's [G87], specifically using $\oplus$ for his par $\parr$; also we use $\top$ as the unit for $\otimes$, $\bot$ as the unit for $\oplus$ (or $\parr$). We use $\times, +, 1, 0$ for the additives, as categorically they are product and coproduct, terminal and initial objects.

"additives", that is categorical products and coproducts, and the "exponential" modalities: **!** "of course" and **?** "why not". Note that having additives is, again, a property rather than structure: a linearly distributive category either has additives or it does not. On the other hand, the exponentials are structure. Semantically they are provided by functors which are appropriate to the categorical doctrine: these are called linear functors and they are actually pairs of functors (*e.g.* product and coproduct for the additives, **!** and **?** for the exponentials) which satisfy certain coherence (equational) requirements. For additives, this coherence structure amounts to the requirement that the tensor distributes over the coproduct and that the "par" distributes over the product. For the exponentials, they must form a linear functor pair which supports duplication. The fact that all the structure of linear logic—including the multiplicatives—may be described in terms of linear functors is one of the remarkable insights gained from the categorical view of its proof theory.

The techniques for studying the proof theoretic structure of fragments of linear logic are also useful in analyzing other categorical—and bicategorical—structures involving monoidal structure. A key tool, introduced by Jean-Yves Girard [G87], was a graphical representation of proofs, "proof nets", to represent the formal derivations or "proofs" of linear logic. The use of graphical languages has now become ubiquitous. The circuit diagrams we shall use here are a graphical representation of *circuits* which have their origin as a term logic for monoidal categories. Circuits are much more generally applicable than Girard's proof nets and provide a bridge between logical and geometric intuitions [JS91]. They are, on the one hand, a formal mathematical language but crucially, at the same time they have an intuitive graphical representation.

As graphical languages have become an almost indispensable tool for visualizing linear logic proofs and, more generally, maps in monoidal categories, we take the time here to describe how "circuits" are formalized and we illustrate their use. A major benefit of using circuits is that they make coherence requirements (*i.e.* which diagrams must commute) very natural. For example, the coherence requirements of linear functors (see section 6.1) are somewhat overwhelming when presented "algebraically", but when seen graphically are very natural.

The use of graphical techniques is also illustrated in our resolution of the coherence problem for $*$-autonomous categories. Here circuits are used to construct free linearly distributive categories: morphisms of these categories are given by circuits, modulo certain equivalences which are generated by graph rewrites. The rewriting system, which is a reduction–expansion system with equalities, is analyzed to produce a notion of normal form, and this allows us to derive a procedure for not only determining the existence of morphisms (between given objects) but also for determining the equality of morphisms (between the same objects).

Some of our techniques and perspectives may seem to lie outside what is traditionally regarded as "proof theory", but they are firmly rooted in the proof theoretic traditions which follow Gentzen's natural deduction. Our approach will be moderately informal, aiming to give the essential ideas involved so the reader may more easily read the technical papers which may be found online.

We wish to dedicate this exposition to the memory of Joachim Lambek. Jim Lambek started the field of categorical proof theory with papers in the 1950s and 60s, and he has been an inspiration for so much of our own work for the past several decades. As a person, Jim will

be missed by all his friends and colleagues; his work will remain a vital influence in all the fields in which he worked.

1.1 Prerequisites

This essay is not entirely self-contained; in particular, we shall assume some familiarity with basics of category theory, of formal logic (especially fragments of linear logic), and with the connections between these (often referred to as the "Curry-Howard isomorphism"). More specifically, the main such assumptions involve the following topics. A refresher on many of these may be found in other papers in this volume, in the references to this essay, and in standard references.

Basic notions of categories. Included in this is the definition of a category (consisting of objects and morphisms or arrows between them, with structure characterizing identity morphisms and composition of morphisms).

Sequent calculus. The reader should be familiar with sequent calculus presentations of logics, specifically how a logic is generated by basic (logical) operations, axioms for these, and deduction rules which specify how the operations operate.

Categorical proof theory. The "equivalence" between objects of a category and well-formed formulas of a logic, and between morphisms of a category and derivations (usually modulo an equivalence relation) will be basic to this essay. (In view of the "Curry-Howard isomorphism", this "equivalence" also extends to the types and terms of a type theory, as may be seen in other chapters in this volume.) This "equivalence" also applies to other categorical structures such as multicategories and polycategories. Our presentation of this equivalence will be "high level", rather than in terms of explicit details. So, for example, when we consider monoidal categories (section 3.1), we regard an object $A \otimes B$ and a "symmetry map" $A \otimes B \longrightarrow B \otimes A$, as a logical formula, rather like a weak notion of conjunction $A \wedge B$, together with a logical entailment $A \wedge B \vdash B \wedge A$ (think of propositional logic, where $A \wedge B$ is logically equivalent to $B \wedge A$). Conversely, given a logical theory, one can construct a category from it as suggested above. When one pays attention to the details, a suitable notion of equivalence may be established between the logical and categorical notions, and so the techniques of one type of structure may be applied to the other type, giving new techniques for the study of each. The details are not necessary for a first reading of the essay, but will be useful for a deeper understanding; such details may be found in many of the references provided.

Graphical representation of logical derivations. There are many ways to denote proofs in a logic: the reader is probably familiar with several, such as Hilbert-style axioms and deduction rules, sequent calculus, natural deduction, and combinators, and may even be familiar with others, for example using the style of programming languages. In this paper we shall present a graphical notation for representing proofs in some simple logics (essentially fragments of linear logic [G87]). The point of such a notation is that

it is particularly well adapted to resolving the types of questions we have about the categorical (and so logical) structures involved, particularly coherence questions (when are two maps equal?, when are two derivations equivalent?), better adapted, in fact, than other presentations of the structure. For example, by using a logical presentation of $*$-autonomous categories, we are able to give a procedure for determining equality of maps (an open question at the time of our paper [BCST96]).

We shall be fairly explicit how derivations may be represented by graphs, but it has to be said that our approach is hardly the only one. There are many variations in such graphical representations: an excellent survey of many used for monoidal categories may be found in [S11]. The main point to stress here is that graphs, called "circuits" in this paper, are used to represent maps in various structured categories, such as monoidal categories, and so (by the "Curry-Howard isomorphism") also derivations (formal proofs) in various logics. However, if these are to be maps in a category with structure, then we must be sure that appropriate equalities of maps hold, and furthermore, if the graphs are also derivations in a logic, such equalities of maps must be coherent with appropriate equivalences of derivations. One reason the use of graphical representations has been so successful is that not only are these equations simply handled, but many such equations appear "for free" (such as the categorical axioms), and moreover different graphical representations have different virtues in this regard—one's preference for one over another usually depends on exactly which is most convenient for the purpose at hand.

Consider the symmetry transformation we mentioned above in the context of monoidal categories: $A \otimes B \to B \otimes A$. It is usual in such a situation to require that applying such a symmetry map twice $A \otimes B \to B \otimes A \to A \otimes B$ should equal the identity map on $A \otimes B$; graphically this amounts to a standard "string rewrite" in our circuit calculus. In a real sense, this is applying the Cartesian connection between geometry and algebra to the logical context.

This connection goes both ways; in [P71], Prawitz wanted to give a notion of "equivalence of proofs", in terms of natural rewriting rules on derivations (in natural deduction); these turned out to be (almost!) exactly the rules needed to capture the corresponding categorical structure (for the $\wedge, \Rightarrow$ fragment, this correspondence was exactly what was needed to capture Cartesian closed categories). Likewise the "string rewrites" needed for logic, and for categorical structure are a natural extension of the Reidemeister moves of string theory (suitably interpreted depending on exactly what logic and categorical structure is involved).

In short, each domain (logic, category theory, and geometry) may have its preferred equations for a structure, and if all goes well (amazingly, it seems to do so for "most natural situations", though it's usually a matter of considerable interest when it doesn't!) these should (more or less) coincide. We'll be considering some significant examples of when they do coincide.

2 Circuits and the basic proof theory of the cut rule—multicategories

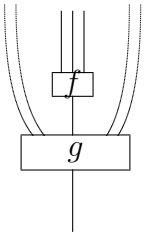
We shall start with the most basic logical system involving a cut rule, namely logic with **only** the cut rule, and its representation in multicategories. The proof theory consists of sequents with a list of formulas on the left of the turnstile, and just one formula on the right (*i.e.* many premises, one conclusion). For logicians, we remark that this means we shall dispense with the usual structural rules of contraction, thinning, and exchange. So the only logical axiom of the system is the identity axiom, and the only other rule is the cut rule. It is useful to name sequents; using Lambek's conventions, these are the deduction rules:

$$\frac{}{1_A\colon A \vdash A}\ ax \qquad\qquad \frac{f\colon \Gamma \vdash A \quad g\colon \Gamma_1, A, \Gamma_2 \vdash B}{g\langle \Gamma_1, f, \Gamma_2\rangle\colon \Gamma_1, \Gamma, \Gamma_2 \vdash B}\ cut$$

When less precision does not cause ambiguity, we shall abbreviate the conclusion of the cut rule as $g\langle f\rangle$, Γ_1 and Γ_2 being understood.

In addition to the identity axioms, we shall allow a set of sequents $f\colon \Gamma \vdash A$ to be used as a starting point for generating proofs: these are often referred to as "non-logical axioms". The proof theory embodied by the two axioms above together with a specified set of non-logical axioms we refer to as a **multi-logic**.

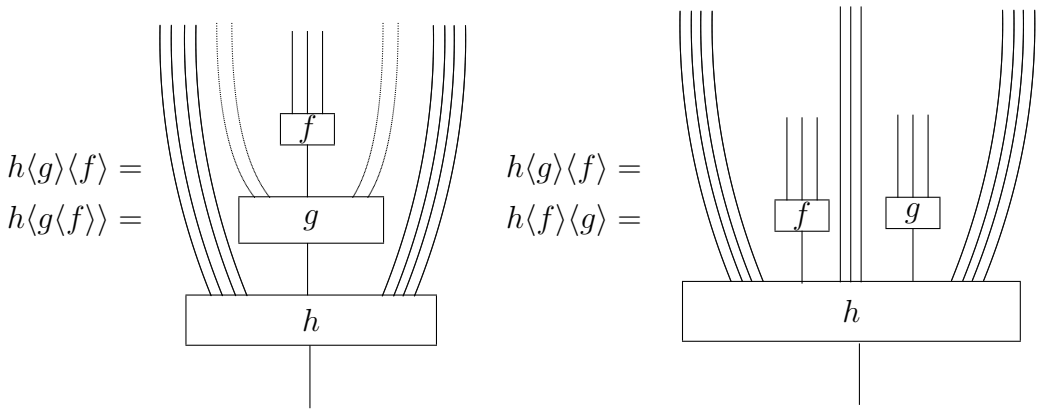
We shall represent sequents graphically with "circuits" whose nodes (shown as boxes) represent sequents, and whose edges or "wires" represent the formulas making up the sequent. The premises of the sequent correspond to the wires entering the box from above, while the conclusion is the wire leaving the box below. So, for example, given non-logical axioms f and g (as above), the cut rule constructs $g\langle \Gamma_1, f, \Gamma_2\rangle$, represented graphically as follows:



A is the type of the output of the top sequent (box) and is connected by a wire of that type to an input of the bottom sequent (box): B is the type of the final output. An axiom sequent is simply represented by the wire corresponding to the formula.

The categorical proof theory of such a logic is Lambek's multicategories [L69]. A multicategory consists of a set of objects and a set of multimorphisms. Each multimorphism has a domain consisting of a list of objects and a codomain consisting of a single object. Each object has an identity multimorphism, whose codomain is the object itself, and whose domain is the singleton list consisting of just that object. Composition is cut, as described above. Appropriate equivalences must be imposed: there are two identity axioms (for "pre-composition" and for "post-composition" of a multimorphism by an identity morphism), and two associativity axioms (when cut is done "vertically" or "horizontally"). Using notation

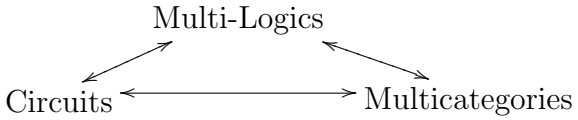
similar to that above, if we are given $f\colon \Gamma \longrightarrow A, g\colon \Gamma_1, A, \Gamma_2 \longrightarrow B, h\colon \Gamma_3, B, \Gamma_4 \longrightarrow C$, then we require $h\langle g\rangle\langle f\rangle = h\langle g\langle f\rangle\rangle$. Also, given $f\colon \Gamma \longrightarrow A, g\colon \Gamma_1 \longrightarrow B, h\colon \Gamma_3, A, \Gamma_4, B, \Gamma_5 \longrightarrow C$, we require $h\langle g\rangle\langle f\rangle = h\langle f\rangle\langle g\rangle$. A nice summary may be found in [L89]. This structure is more intuitively presented by the circuit diagrams. The two identity axioms, pictorially represented, merely amount to noticing that extending a wire with an identical (*i.e.* an identity) wire does not change the circuit. The two associativity axioms merely assert that the two "obvious" ways to "compose" two diagrams produce the same circuit. To illustrate this, here are the two circuits resulting from the two associativity axioms.



The appropriate notion of morphism of multicategories, called multifunctor, preserves the composition and the identities. This corresponds to an interpretation of one multi-logic into another.

A multi-logic is determined by its presentation as propositions and non-logical axioms: these determine a "multi-graph". However, given a multi-graph, that is, a set of objects and a set of "multi-arrows" (arrows whose domain is a list of objects, and whose codomain is a single object), it is clear how we can construct a multicategory which is generated by the multi-graph. One closes the multi-arrows under composition (equivalently cut), and one factors out by the equivalences required of a multicategory. Thus, given a presentation of a multi-logic, we may generate a multicategory. It is clear the construction outlined above will provides the free multicategory generated by the multi-graph corresponding to the non-logical axioms. This is the categorical semantics of the multi-logic.

Indeed, we have an equivalence of categories (2-categories in fact) between basic cut logics (generated by non-logical axioms with proof identifications) and multicategories (based on non-logical axioms). Thus we have constructed the following "triangle of doctrines", where the two-headed arrows represent equivalences of categories:



Next we shall add connectives to this logic, and explain the corresponding categorical notions, features, and circuits.

3 ⊗-multi-logic, representable multicategories, and monoidal categories

We now pass to a simple categorical structure, the free monoidal category generated by some objects and morphisms, in order to understand the effect of introducing connectives into a multi-logic. In effect, we shall be "representing" the commas in the sequents of multi-logic by connectives.

In our discussion of multi-logic and multicategories, we allowed our sequents to have lists of premises: this means that the order in which the premises occurred is important. Often, however, we will want to consider logics in which the order of premises does not matter. This means that the premises may be viewed as "bags", or "multisets" rather than lists. Logically this is accommodated by the addition of the "exchange rule":

$$\frac{\Gamma_1, A, B, \Gamma_2 \vdash C}{\Gamma_1, B, A, \Gamma_2 \vdash C}\ exch$$

which permits neighbouring premises to be swapped. In circuits, this corresponds to allowing wires to cross so the circuits are no longer "planar".

Often one refers to a (multi-)logic in which the order of premises matters as a "non-commutative" logic and one in which the exchange law is present as a "commutative" logic. On the categorical side one refers to multicategories in which there is no exchange rule as being "non-symmetric" and those in which the exchange rule (crossing wires) is allowed as being "symmetric".

3.1 Monoidal categories

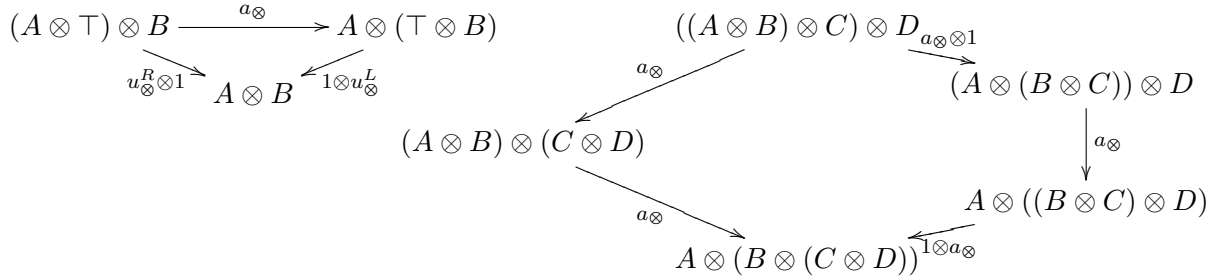
We begin by fixing notation, recalling the definition of a monoidal category.

Definition 3.1 (Monoidal categories) *A monoidal category $\langle \mathbf{C}, \otimes, \top \rangle$ consists of a category $\mathbf{C}$ with an associative bifunctor (a "tensor") $\otimes$ with a unit $\top$. If the tensor is symmetric, we shall refer to $\mathbf{C}$ as a symmetric monoidal category.*

To say the tensor has a unit, and is associative means that we have the following natural isomorphisms.

$$u^R_\otimes : A \otimes \top \to A \qquad u^L_\otimes : \top \otimes A \to A \qquad a_\otimes : (A \otimes B) \otimes C \to A \otimes (B \otimes C)$$

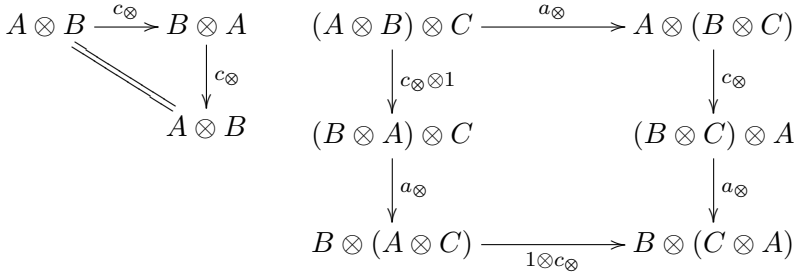
which must satisfy the following coherence equations (expressed as commuting diagrams):



The tensor is symmetric when there is, in addition, the following natural isomorphism:

$$c_\otimes \colon A \otimes B \to B \otimes A$$

which must satisfy the following coherence requirements:



One may think of the tensor as representing a (weak) notion of conjunction ("and"), but this is a conjunction without the structural rules of contraction and thinning, and in the nonsymmetric case, without exchange as well.

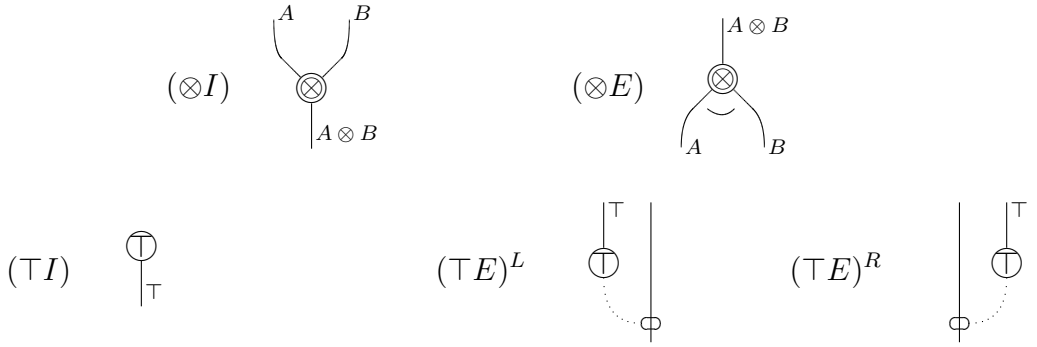
3.2 Sequent calculus and circuits for monoidal categories

The sequent calculus presentation of the logic of monoidal categories adds to basic cut logic the tensor $\otimes$ as a logical operator, together with a unit $\top$ for the tensor (generating well founded formulas in the usual way, so that if A, B are well founded formulas, so is $A \otimes B$, as are atomic formulas and $\top$), as well as the following rules:

$$(\otimes R)\ \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \qquad (\otimes L)\ \frac{\Gamma_1, A, B, \Gamma_2 \vdash C}{\Gamma_1, A \otimes B, \Gamma_2 \vdash C}$$

$$(\top R)\ \frac{}{\vdash \top} \qquad (\top L)\ \frac{\Gamma_1, \Gamma_2 \vdash C}{\Gamma_1, \top, \Gamma_2 \vdash C}$$

Corresponding to these rules, we enrich our circuits with nodes for the tensor and its unit: we have "tensor introduction" and "tensor elimination" nodes, "unit introduction" and "unit elimination" nodes, as well as any non-logical axioms we may have assumed (which now may involve composite types or formulas involving tensor). The introduction and elimination nodes look like this.



The (optional) small arc drawn below the $(\otimes E)$ node indicates that this node is of a different nature from its $(\otimes I)$ cousin—in part this reflects that the introduction node represents a valid deduction $A, B \vdash A \otimes B$ of the logic, whereas the elimination node does not (as $A \otimes B \vdash A, B$ is not a valid multi-sequent). We call the elimination node a "switching link"; it represents the simultaneous existence of two wires whose outputs are tensored together. We have placed a restriction on non-logical components, that they have only one output wire, but that restriction could be replaced with a restriction that the output wires of a non-logical component be switching. Using the introduction link, one can tensor together output wires so as to have a single output wire in their place. Note that in the present monoidal case, any node that has several output wires must be switching (at the output wires), and no node has switching wires at the input. The idea of "switching" will be particularly relevant when we consider poly-logics, as there which wires are "switching" becomes a matter of greater subtlety and importance.

The curved "lasso"-like wires, called thinning links, used in the unit nodes are rather different from the other components, especially in the more general cases soon to be considered, and so are denoted with dotted lines. The reader can consider the loop at the end of the lasso as a movable node and the lasso itself as a wire. The unit nodes correspond to (derivable) sequents $\vdash \top$, $\top, A \vdash A$, and $A, \top \vdash A$.

3.2.1 Representability

Sequents $A \vdash B$ are derivable in this sequent calculus if and only if there is a corresponding circuit with one input wire of type A, and one output wire of type B. Any valid circuit can be "represented" by such a one-in-one-out circuit by tensoring all the inputs wires together and tensoring all the output wires together. These one-in-one-out sequents/circuits then correspond to morphisms in a monoidal category.

Thinking in terms of natural deduction, the sequent rules induce bijective correspondences indicated by these "rules":

$$\frac{\Gamma_1, A \otimes B, \Gamma_2 \vdash C}{\Gamma_1, A, B, \Gamma_2 \vdash C} \qquad \frac{\Gamma_1, \top, \Gamma_2 \vdash C}{\Gamma_1, \Gamma_2 \vdash C}$$

This sets up a natural bijection between proofs and multi-arrows in a "representable multi-category" [H00]. Multi-arrows then correspond bijectively to one-in-one-out sequents, and thus to maps in a tensor category. To show that these do correspond properly, the simplest route is *via* the circuits.

3.3 Circuit rewrites

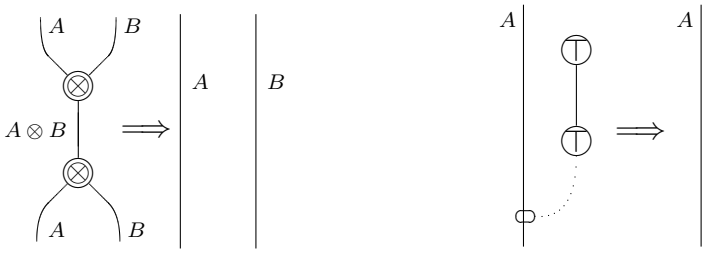
Of course, merely having circuits is just the start of the matter. We want circuits to not only correspond to proofs in the logic but also to have the structure of a monoidal category. In the present simplified case, any circuit with just one input and one output always represents

a morphism in a monoidal category. In general, we should expect there to be a more complicated "correctness" condition on circuits which corresponds to how proofs in the logic (or maps in the categorical doctrine) are constructed. Shortly we shall meet such a correctness condition for poly-logics.

A question, arising not only from categorical considerations, but also from logic [P71], is: when are two morphisms or proofs equivalent? This is in some sense *the* fundamental question for the logics we are discussing. It turns out that circuits are a very convenient tool for resolving this question. We shall now focus on this question.

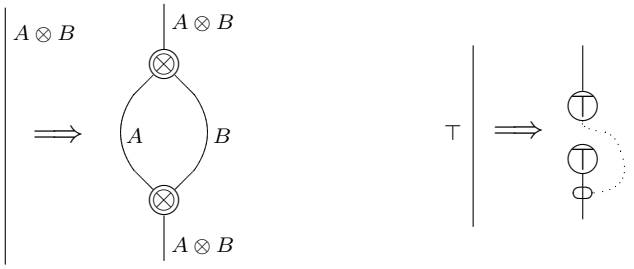
There is an equivalence relation on circuits, generated by the following rewrites which we present as a reduction–expansion system modulo equalities.

Reductions:



(There is a mirror image rewrite for the unit, with the unit edge and nodes on the other side of the A edge.)

Expansions:



(Again, there is a mirror image rewrite for the unit, with the thinning edge on the other side of the unit edge and node.)

In addition to these rewrites, there are also a number of equivalences, see Figure 1, which must be imposed to account for the unit isomorphisms. In effect, one is allowed to move a unit lasso past any subcircuit that corresponds to a valid derivation, provided that validity is preserved. At this basic monoidal level, these equivalences actually allow the unit lasso to be moved onto *any* wire—provided a circuit is produced. For this reason, when dealing with this level of logic the thinning links will often be omitted. However, we shall shortly see why thinning links cannot be omitted in the more general setting we consider.

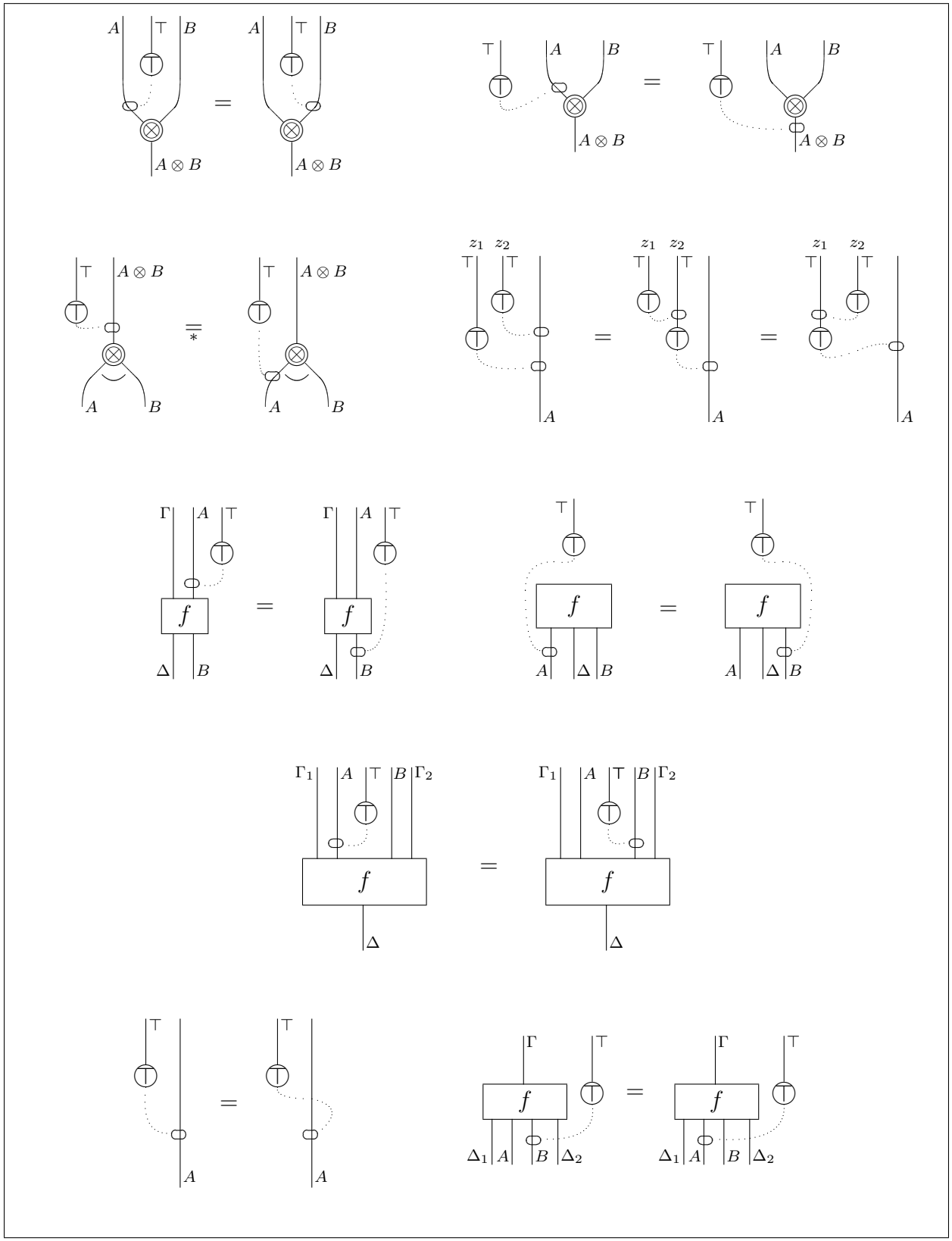


Figure 1: Unit rewirings

3.3.1 Normal Forms

Given the system of reductions and expansions, modulo equations, as described above, we can show that the reduction and expansion rewrites terminate, that there is a Church-Rosser theorem, modulo equations, and so there is a notion of expanded normal form, again modulo equations, for proof circuits. Essentially, the shape of a normal form involves tensor elimination steps (at the top), with some rearrangement of wires (in the middle), ending with tensor introductions (at the bottom).

In the present case, all this is not too difficult to see. First notice that the reductions always remove material and so must certainly terminate. In contrast, the expansions always introduce material as the idea is that they "express" the type the wire they expand. One can imagine repeatedly expanding a wire along its length: this means that the expansions do not terminate in the usual sense. However, in an expansion/reduction system, expansions which can be immediately removed by reduction have no net effect on the system. Thus, an expansion/reduction system works by reducing terms to reduced form and using the expansions to facilitate moves between reduced forms (this is a sort of annealing process). This means one only applies expansion rules to reduced terms and, having applied the expansion, one immediately reduces the result. An expanded normal form is a reduced circuit for which all expansions have no net effect after the reductions are reapplied.

Returning to the case of expanding a wire twice along its length, it is easy to see that the second expansion will immediately trigger a reduction back to the single expansion. Thus, a wire can only be expanded at most once on its length. A wire of composite type when expanded produces wires having strictly smaller types which can in turn be expanded in a nested fashion. However, this sort of expansion nesting must terminate as the types of the produced wires become strictly smaller. This shows that this expansion reduction system terminates.

Two examples of rewriting are shown in Figure 2 and Figure 3. The former shows Mac Lane's pentagon coherence condition $a_{\otimes}; a_{\otimes} = a_{\otimes} \otimes 1; a_{\otimes}; 1 \otimes a_{\otimes}$: the left-hand side and the right-hand side of the equation are shown on the far left and far right. In the middle, the common reduction shows they are equal. In this example, repeated use of the tensor reduction rule is all that is needed. To illustrate the need for the unit equivalence and the lassos, in Figure 3 we show how the unit coherence condition ($a_{\otimes}; 1 \otimes u^L_{\otimes} = u^R_{\otimes} \otimes 1$) can be proven using circuits.

3.4 Summary so far

The proofs of a multi-logic may be presented as multi-arrows in a multicategory. These, in turn can be represented using circuit diagrams: in Section 5 we will provide a more formal treatment of circuits. Representable multicategories are monoidal categories and these correspond to $\otimes$-multi-logics. $\otimes$-multi-logics have a cut elimination theorem. Viewed as a rewriting system on circuits this becomes an expansion/reduction rewriting system which allows one to decide equality of proofs. Finally, the equivalence classes of circuits (or of derivations) are morphisms of a category of circuits, and this is the free (symmetric) monoidal category (over a generating multi-graph of components).

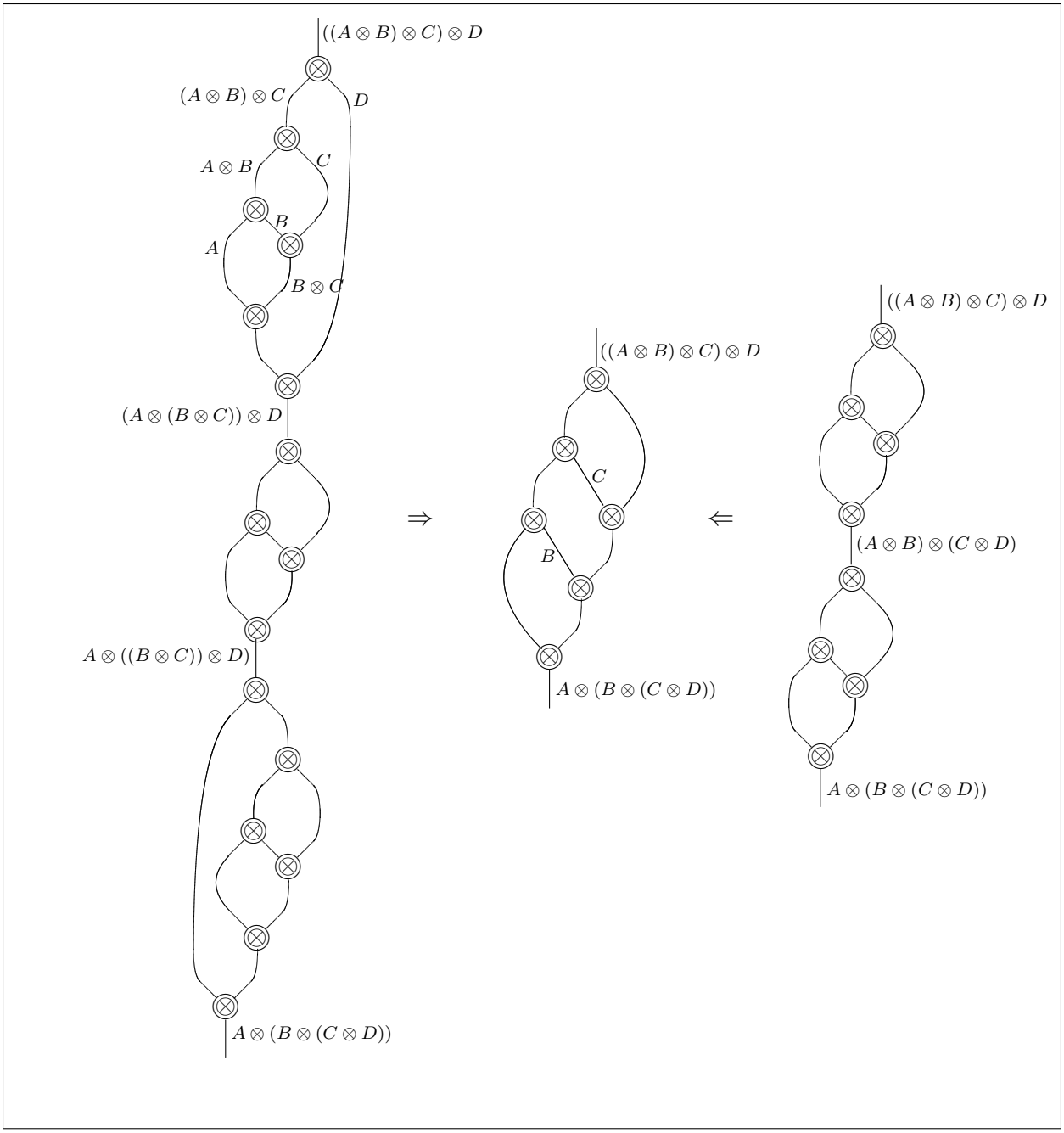


Figure 2: Pentagon coherence condition

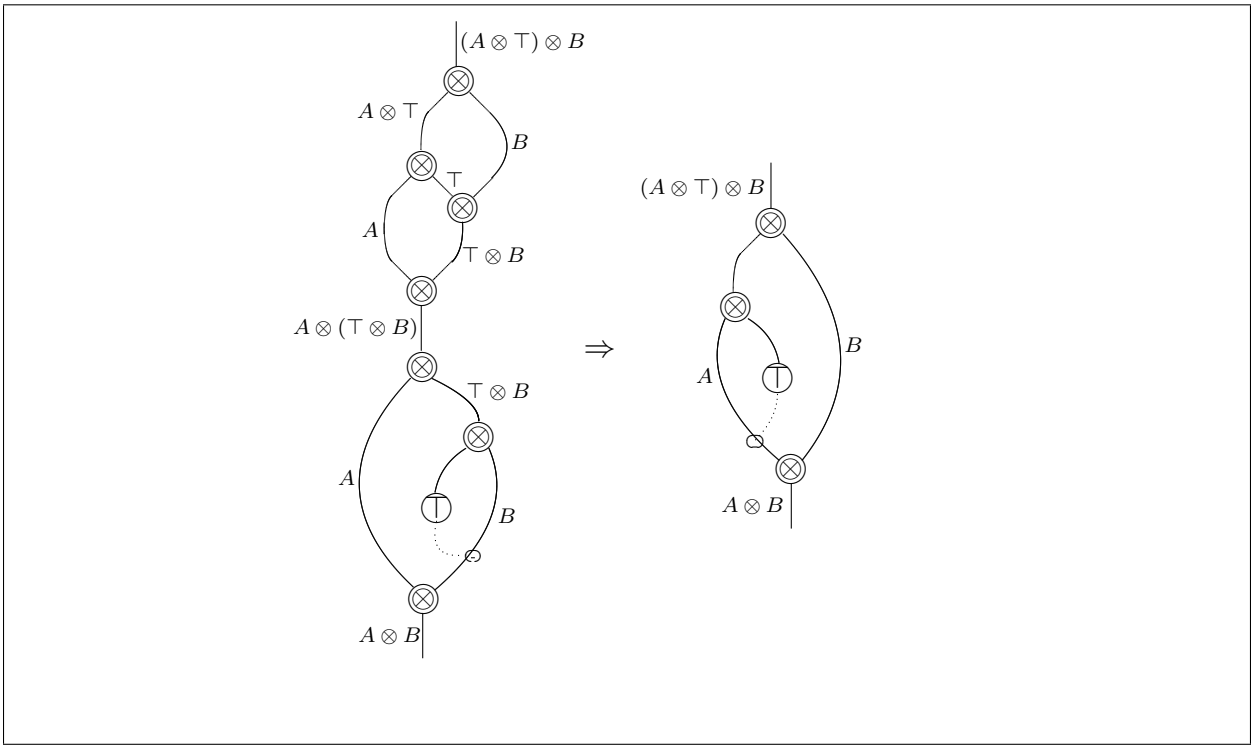
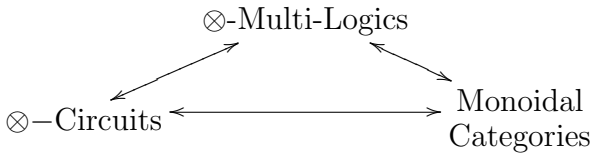


Figure 3: Unit coherence condition

We have already seen that circuits form a (symmetric) monoidal category where the objects are formulas and morphisms are equivalence classes of circuits. That it is the free such is the force of Mac Lane's coherence theorem; a proof of these claims (in the linearly distributive context) may be found in [BCST96]. This may be summarized by the following conceptual diagram (which actually represents 2-equivalences between appropriate chosen 2-categories).



4 Tensor and Par: basic components of linear logic

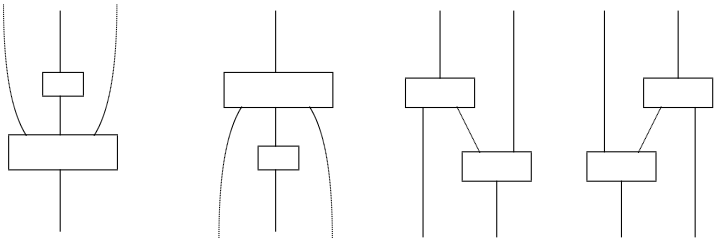
Imagine setting up a poly-logic: its sequents permit many (or no) premises on the left of the turnstile and many (or no) conclusions on the right of the turnstile. For the moment assume no commutativity: this means the cut rule has to take several forms which gives the following presentation as a sequent calculus.

$$\frac{}{A \vdash A}\ id$$

$$\frac{\Gamma \vdash A \quad \Gamma_1, A, \Gamma_2 \vdash \Delta}{\Gamma_1, \Gamma, \Gamma_2 \vdash \Delta}\ cut_1 \qquad \frac{\Gamma \vdash \Delta_1, A, \Delta_2 \quad A \vdash \Delta}{\Gamma \vdash \Delta_1, \Delta, \Delta_2}\ cut_2$$

$$\frac{\Gamma_1 \vdash \Delta_1, A \quad A, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ cut_3 \qquad \frac{\Gamma_1 \vdash A, \Delta_1 \quad \Gamma_2, A \vdash \Delta_2}{\Gamma_2, \Gamma_1 \vdash \Delta_2, \Delta_1}\ cut_4$$

One might have expected the identity axiom $A \vdash A$ to have had a more general form as $\Gamma \vdash \Gamma$. However, that would correspond to what is known as the "mix" rule, and would gives the logic quite a distinct and different flavour. Among other things, the mix rule would allow a circuit to have several disconnected components; without mix, circuits have to be connected. The circuits corresponding to the four cut rules are drawn below. Note that for simplicity, we have represented lists of formulas, such as Γ, as well as single formulas, such as A, by single wires. In each case, the only wires that *must* correspond to single formulas are those that join the two boxes being cut.



The reader should notice that any circuit inductively built from these rules, that is a **poly-circuit**, not only must be connected but also cannot have any cycles (*i.e.* must be acyclic). This, in fact, is precisely the *correctness criterion* for poly-circuits.

This circuit calculus is the basis for the categorical structure of a polycategory, as discussed for example in [CS97a, CKS03]: in fact, circuits over an arbitrary poly-graph, with their natural notion of equivalence and satisfying the correctness criterion, form the free polycategory over that poly-graph.

4.1 $\otimes\oplus$-poly-logic

To capture the appropriate categorical notion, we must add appropriate tensor structures so that the "commas" of poly-logic. on both sides of the poly-sequents, are represented. The commas on the left of the turnstile are interpreted differently from those on the right. Thus there are two connectives: $\otimes$ ("tensor") for commas on the left and $\oplus$ ("par") for commas on the right. The behaviour of these connectives is determined by requiring that polycategory be representable in the sense that there are the following bijective correspondences between poly-maps:

$$\frac{\Gamma_1, \Gamma_2 \vdash \Delta}{\Gamma_1, \top, \Gamma_2 \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta_1, \Delta_2}{\Gamma \vdash \Delta_1, \bot, \Delta_2}$$

$$\frac{\Gamma_1, X, Y, \Gamma_2 \vdash \Delta}{\Gamma_1, X \otimes Y, \Gamma_2 \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta_1, X, Y, \Delta_2}{\Gamma \vdash \Delta_1, X \oplus Y, \Delta_2}$$

Translating this into a sequent calculus presentation gives the sequent calculus presentation of $\otimes\oplus$**-poly-logic**: this consists of the cut rules of poly-logic, as above, together with the

following logical rules governing these new connectives:

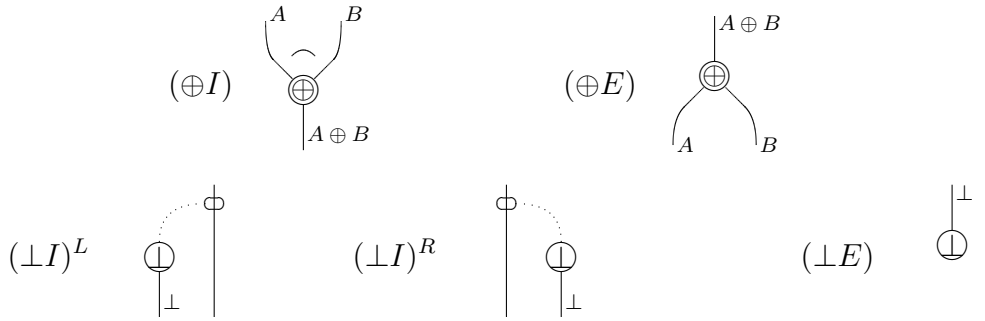
$$\frac{\Gamma_1, \Gamma_2 \vdash \Delta}{\Gamma_1, \top, \Gamma_2 \vdash \Delta}\ (\top L) \qquad \frac{}{\vdash \top}\ (\top R)$$

$$\frac{}{\bot \vdash}\ (\bot L) \qquad \frac{\Gamma \vdash \Delta_1, \Delta_2}{\Gamma \vdash \Delta_1, \bot, \Delta_2}\ (\bot R)$$

$$\frac{\Gamma_1, X, Y, \Gamma_2 \vdash \Delta}{\Gamma_1, X \otimes Y, \Gamma_2 \vdash \Delta}\ (\otimes L) \qquad \frac{\Gamma_1 \vdash \Delta_1, X \quad \Gamma_2 \vdash Y, \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, X \otimes Y, \Delta_2}\ (\otimes R)$$

$$\frac{\Gamma_1, X \vdash \Delta_1 \quad Y, \Gamma_2 \vdash \Delta_2}{\Gamma_1, X \oplus Y, \Gamma_2 \vdash \Delta_1, \Delta_2}\ (\oplus L) \qquad \frac{\Gamma \vdash \Delta_1, X, Y, \Delta_2}{\Gamma \vdash \Delta_1, X \oplus Y, \Delta_2}\ (\oplus R)$$

The circuits that correspond to these rules are induced by the following basic nodes: the familiar tensor and tensor unit nodes from the tensor circuits, and dual nodes for the par and par unit $\bot$.



A proof of the $\otimes\oplus$-poly-logic may be represented by a circuit using these components However, not every circuit made of these components represents a proof! Thus, to guarantee that a circuit does represent a proof, there is a correctness criterion which must be satisfied. One way to express this correctness criterion, due to Girard, is by a "switching condition": a circuit satisfies this condition if, whichever way one sets the "switch" in each switching link, the circuit remains connected and acyclic. To set the switch in a switching link one disconnects one of the two wires between which the "switch" links. A circuit satisfying this correctness criterion (and thus represents a proof in $\otimes\oplus$-poly-logic) is a $\otimes\oplus$-circuit.

Notice that using this criterion one can demonstrate that a circuit *does not* satisfy the correctness criterion by a non-deterministic polynomial time (NP) algorithm. This algorithm works by guessing a configuration of the switches and proving the circuit with that set of disconnections is either cyclic or disconnected. Providing a configuration of switches to witness the *incorrectness* of a circuit is a very effective way of showing the invalidity of a circuit. However, to show that a circuit satisfies the correctness criterion one must potentially try all possible switching configurations ... and there are exponentially many of these. Thus, algorithmically this is somewhat disastrous!

Fortunately, [DR89] describes a linear time algorithm for the correctness criterion which is based on more directly checking that a circuit represents a proof. This correctness criterion can easily be adapted for $\otimes\oplus$-circuits and this is described in [BCST96].

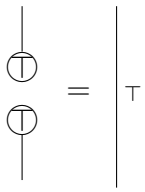
4.2 Why thinning links?

A curiosity is the apparent lack of symmetry between unit introduction and elimination links. Logically they correspond to the (bijective) correspondences we saw above. We might have expected the $\top$ elimination link ($\top I$) and the $\bot$ elimination link ($\bot E$) to be simpler without a lasso. But this simply does not work! As this is a rather crucial aspect of our circuit calculus, some discussion of this is in order.

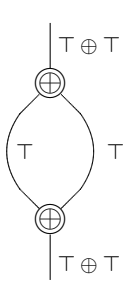
We shall want circuit identities corresponding to the equivalence of the following proof (which uses a cut on the left $\top$) and the identity axiom:

$$\dfrac{\vdash \top \qquad \dfrac{\top \vdash \top}{\top, \top \vdash \top}}{\top \vdash \top}$$

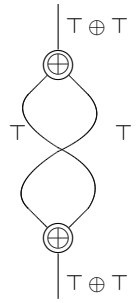
(and dually for $\bot$). If we had let the ($\top E$) link be without lasso, as suggested above, this identity would become:



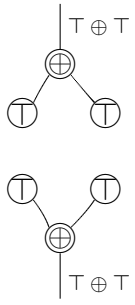
This will not do, however—the lack of a thinning link here is fatal to the coherence questions which concern us. To see why, consider the following simple example which compares the identity with the par twist map applied to the tensor unit "par"ed with itself:



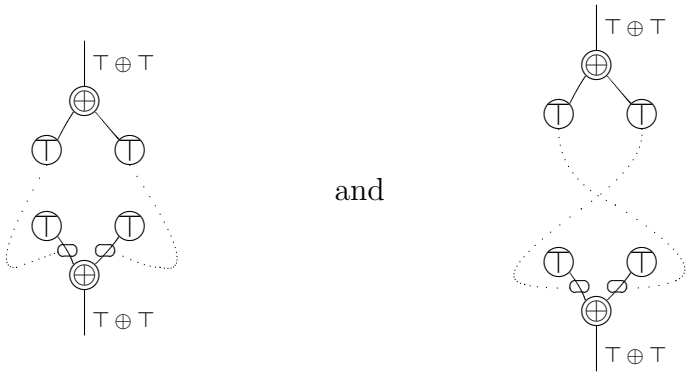
and



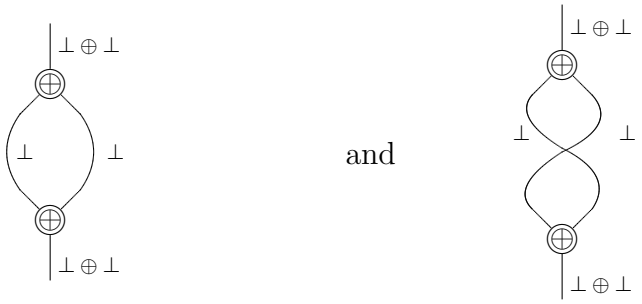
Given the above identity without thinning links these would both be equivalent to the same net as expanding the wires of type $\top$ would lose the twist because the circuit becomes disconnected.



The twist and the identity on $\top \oplus \top$ are not equivalent as morphisms. For example, **PointedSets** is a linearly distributive category with product and coproduct as tensor and par ([CS97a, Proposition 3.4], but in this category clearly the twist on the two element set is not the identity. The point is with thinning links we can at least distinguish these maps as nets, as we see below:



so there is hope that we can arrange for them to be inequivalent (and indeed they are). Note, however, the different behaviour of the units: if we replace $\top$ with $\bot$, then these nets do correspond to equivalent derivations, since $\bot \oplus \bot$ is isomorphic to $\bot$ and the identity is the same as the twist on $\bot$. Thus, these two nets when we expand the $\bot$ identity wires in the same manner must be equivalent.



To make these circuits equivalent it is clear that we must be able to rewire the thinning links in some manner, but equally not all rewirings can be permissible, so as to keep the distinctions seen with the two maps $\top \oplus \top \longrightarrow \top \oplus \top$ above. It is not too surprising that rewirings will be required: thinning links merely indicate a point at which a unit (or counit) has been introduced by thinning and there is considerable inessential choice going on here. For example, consider the three sequent calculus derivations of the sequent $A, \top, B \longrightarrow A \otimes B$ obtained by thinning in each of the possible places (these clearly should be equivalent):

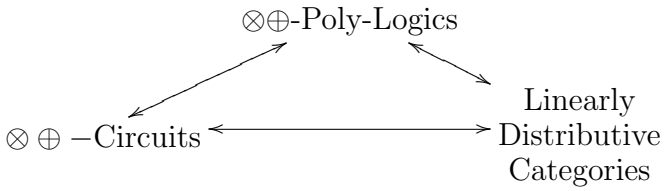
$$\dfrac{\dfrac{A \longrightarrow A \quad B \longrightarrow B}{A, B \longrightarrow A \otimes B}}{A, \top, B \longrightarrow A \otimes B} \quad = \quad \dfrac{\dfrac{A \longrightarrow A}{A, \top \longrightarrow A} \quad B \longrightarrow B}{A, \top, B \longrightarrow A \otimes B} \quad = \quad \dfrac{A \longrightarrow A \quad \dfrac{B \longrightarrow B}{\top, B \longrightarrow B}}{A, \top, B \longrightarrow A \otimes B}$$

As circuits, these are just the $(\otimes I)$ node with a $(\top E)$ link attached to the three possible links. It turns out that the allowable rewirings are essentially those from Figure 1, their obvious

"duals" involving $\bot$ and $\oplus$, and a few involving interactions between these two structures [BCST96].

4.3 Linearly distributive categories

To introduce linearly distributive categories, consider the following conceptual diagram, recalling the connection between tensor circuit diagrams, multicategories, and monoidal categories. This states the analogous connection between the various structures for two tensors should also hold. The intention behind the definition of a linearly distributive category is, thus, that it should be viewed as the (one-in-one-out) maps of a "representable polycategories".



A linearly distributive category has two tensors: the "tensor" $(\otimes, a_\otimes, u^L_\otimes, u^R_\otimes)$, and the "par" $(\oplus, a_\oplus, u^L_\oplus, u^R_\oplus)$ both satisfying the usual coherences of a tensor. The interaction of these tensors is mediated (in the non-commutative case) by two natural linear distribution maps:

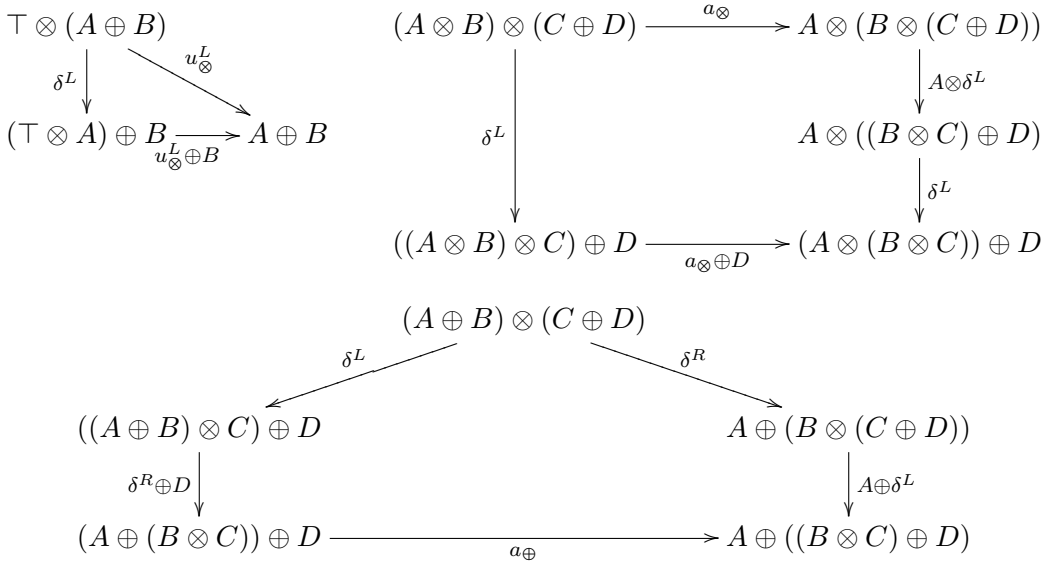
$$\delta^L\colon A \otimes (B \oplus C) \longrightarrow (A \otimes B) \oplus C \qquad \delta_R\colon (B \oplus C) \otimes A \longrightarrow B \oplus (C \otimes A)$$

These must satisfy a number of coherences which, categorically, may be seen as (linear) strength coherences. Before discussing these, however, it is worth understanding the manner in which the linear distributions arise from the interaction of having representation for the commas and the behaviour of the cut. The following derivation of δ^L demonstrates this interaction:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B \oplus C \vdash B \oplus C}\,\mathrm{id}}{B \oplus C \vdash B, C}\,\oplus R \quad \dfrac{\dfrac{}{A \otimes B \vdash A \otimes B}\,\mathrm{id}}{A, B \vdash A \otimes B}\,\otimes L}{A, B \oplus C \vdash A \otimes B, C}\,\mathrm{Cut}}{A \otimes (B \oplus C) \vdash (A \otimes B) \oplus C}\,\otimes L, \oplus R$$

The definition of a linearly distributive category is subject to a number of symmetries these arise from reversing the tensor ($A \otimes B \mapsto B \otimes A$), reversing the par ($A \oplus B \mapsto B \oplus A$) and reversing the arrows themselves while simultaneously swapping tensor and par (thus $\delta^L\colon A \otimes (B \oplus C) \longrightarrow (A \otimes B) \oplus C$ becomes $\delta^R\colon (A \oplus B) \otimes C \longrightarrow A \oplus (B \otimes C)$). We present three coherence diagrams which are complete in the sense that together with these symmetries

they can generate all the coherences for (non-symmetric) linearly distributive categories:

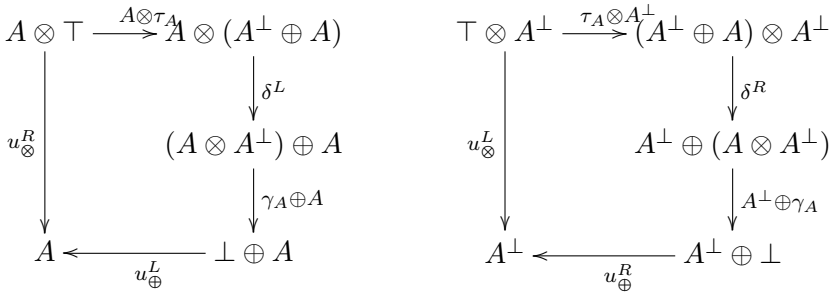


4.3.1 Negation

A key ingredient in the original account of linear logic, which is missing in linearly distributive categories, is negation. So just what do we need to obtain negation in a linearly distributive category? First, we need a function (on objects), which we shall denote $A \mapsto A^\perp$. In the non-symmetric case we shall need two such object functions $A \mapsto A^\perp$ and $A \mapsto {}^\perp A$; for simplicity, we shall outline the symmetric case here, where ${}^\perp A = A^\perp$. Also we need two parametrized families of maps[4]:

$$A \otimes A^\perp \xrightarrow{\gamma_A} \perp \qquad\qquad \top \xrightarrow{\tau_A} A^\perp \oplus A$$

which satisfy the following coherence conditions

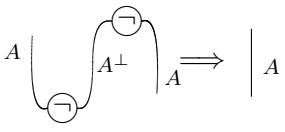


As circuits, this is as follows. The links are represented as "bends":

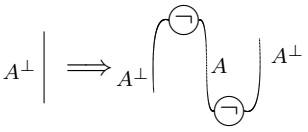
[4]We do not assume any naturality for these maps. In fact, they turn out to be dinatural transformations.

And the equivalences (rewrites) are these:

(¬ Reduction)



(¬ Expansion)



Of course, the hope is that the function $A \mapsto A^\perp$ is a contravariant functor, and the families of maps are dinatural transformations: this is indeed the case. In fact, the category of $\otimes\oplus$-circuits with negation (in this sense) generated from a poly-graph is the free $*$-autonomous category generated by said poly-graph. The 2-category of linearly distributive categories with negation and linear functors (which we shall discuss shortly) is equivalent to the category of $*$-autonomous categories with monoidal functors. Moreover, there is a conservative extension result, stating that the extension of the tensor-par fragment of linear logic to full multiplicative linear logic (which includes negation) is conservative; more precisely, the functor from the category of linearly distributive categories to the category of $*$-autonomous categories extends to an adjunction (the right adjoint being the forgetful functor), whose unit is full and faithful, and whose counit is an equivalence [BCST96].

One consequence of this is that we now have a good circuit calculus for $*$-autonomous categories—the categorical doctrine corresponding to multiplicative fragment of linear logic—and this allows us to give a decision procedure for equality of maps in these categories. Bear in mind that this decision procedure necessarily involves a search as this is a PSPACE complete problem [HH14] and so is not very efficient! As an example of this at work, here is a classic problem (not completely solved until [BCST96]), usually called the triple-dual diagram:

$$\begin{array}{ccc} & ((A \multimap I) \multimap I) \multimap I & \\ {\scriptstyle 1}\swarrow & & \searrow{\scriptstyle k_A \multimap 1} \\ ((A \multimap I) \multimap I) \multimap I & \xleftarrow[k_{A\multimap 1}]{} & (A \multimap I) \end{array}$$

(using $k_A : A \longrightarrow ((A \multimap I) \multimap I)$, the exponential transpose of evaluation.)

In $*$-autonomous categories, (or monoidal closed categories), this diagram generally does not commute. This is easy to see if I is not a unit. If I is a unit, then the diagram does commute if $I = \bot$, generally does not commute if $I = \top$, but does commute if $A = I = \top$.

We note that such instances of this diagram in fact can be done in the linearly distributive context, if we define the internal hom as a derived operation: $A \multimap B = A^\perp \oplus B$, replace I with a unit and $I^\perp$ with the other unit, and replace the negation links with appropriate derived rules corresponding to the (iso)morphisms $\top \otimes \bot \longrightarrow \bot$ and $\top \longrightarrow \bot \oplus \top$. Then we translate the composite $k_{A\multimap 1}; k_A \multimap 1$ above into a proof net: the left side of Figure 4 is a step on the way to its expanded normal form

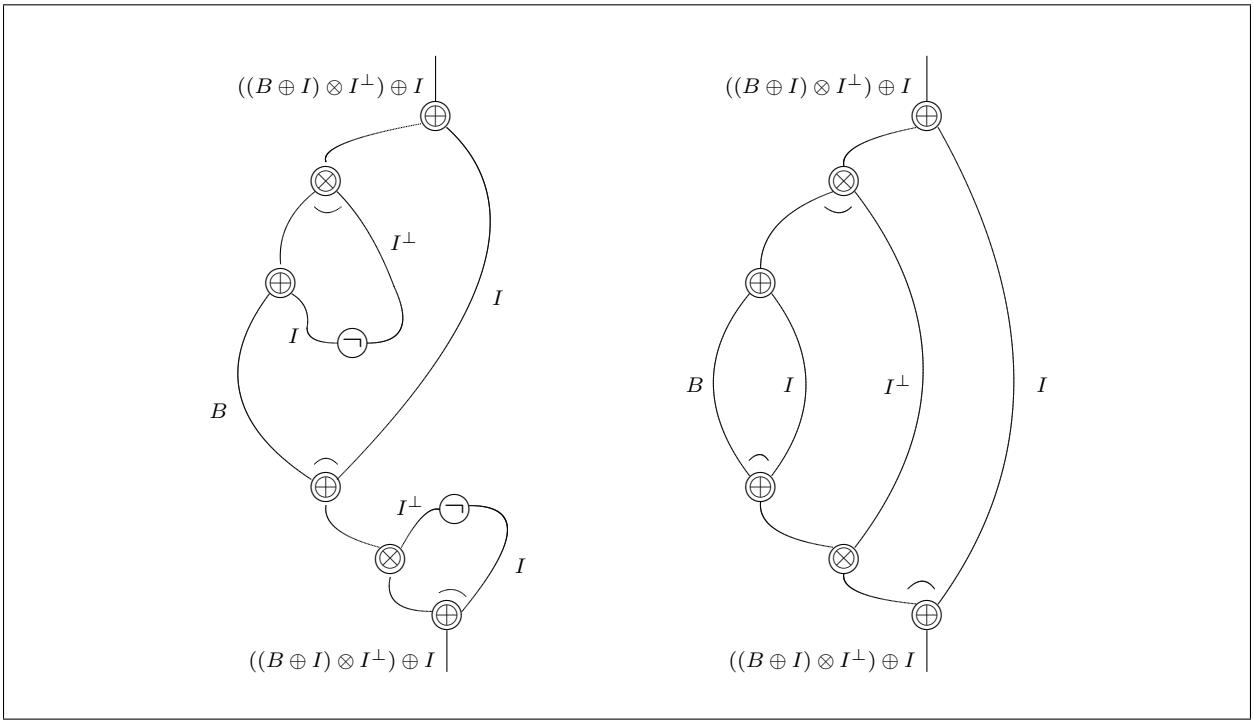


Figure 4: Two sides of the triple-dual triangle

(we write B for $A^\perp$ to prepare for the version of the net that may be constructed in the linearly distributive context). The right side is a similar step in calculating the expanded normal form of the identity circuit.

In the circuits of Figure 4, if I were not a unit, these would be the expanded normal forms, and clearly these nets are not the same. An old idea of Lambek's [L69] may be seen here: the "generality" of the first net is clearly a derivation of the sequent $((B \oplus C) \otimes C^\perp) \oplus D \to ((B \oplus D) \otimes E^\perp) \oplus E$, whereas the generality of the second is $((B \oplus C) \otimes D) \oplus E \to ((B \oplus C) \otimes D) \oplus E$. This is no surprise; it is exactly what one would expect if I was not a unit. Next consider the case if I is a unit.

If $I = \top$, say, then the nodes at I and $I^\perp$ have to be expanded, since in expanded normal form, each occurrence of a unit (recall $\top^\perp = \bot$) must either come from or go to a null node. This in effect transforms several of the edges in the graphs above into thinning links. We leave it as an exercise to show that in this case no rewiring is possible; hence the diagram does not commute. And similarly that the rewiring may be done if $I = \bot$, so that diagram does commute.

But now consider the case where $A = I = \top$ (where the diagram commutes). We must show how to rewire the net corresponding to the compound morphism to give the identity. This is shown in Figure 5: the point here is that with $A = \top$ we have an extra unit and thinning link (corresponding to the wire/thinning link for $\bot$ replacing B at the left), which has a possible rewiring. Although it is not immediately obvious why this thinning link should be rewired, doing so makes other rewirings possible, and once the required rewirings are done, the initial rewiring is reversed to finish with the expanded normal form of the identity map.

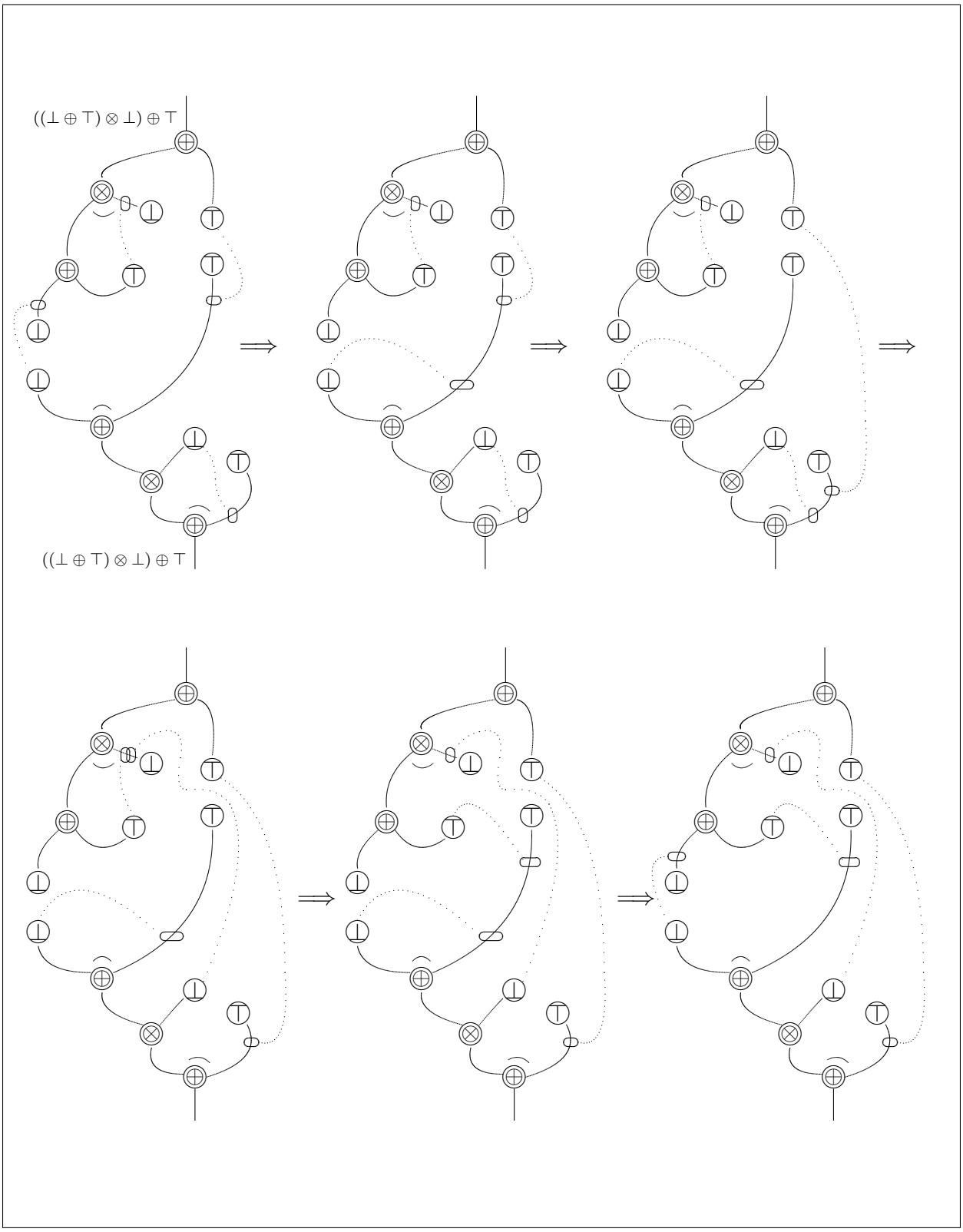


Figure 5: Rewiring the triple-dual

5 Circuits

While the circuits diagrams provide a convenient representation of morphisms or proofs, they have a formal underpinning which is also of interest. Although it is tempting to conflate the two notions, a circuit diagram is, in fact, a diagrammatic representation of something more fundamental, which we call a *circuit*. Circuits provide a term logic for monoidal settings. Such term logics have a distinguished pedigree, going back to Einstein's summation notation for vector calculations, and including diagram notations by Feynman and by Penrose, for example. Their notations were more concretely attached to the specifics of the vector space contexts they intended. Our circuits—which are derived not only from Girard's proof nets but also from the work of Joyal and Street [JS91]—are intended to be more general. Vector space manipulations are a canonical example of tensor category manipulations, so that there is a direct ancestry is not a surprise. However, unlike our predecessors, including Joyal and Street, we explicitly intended our notation to be a term logic, and in particular we applied it to solve the coherence problems associated with linearly distributive categories. Despite this, the term logic was initially invented to facilitate calculations in monoidal categories, and so we shall return to this more straightforward application in the present essay. To extend these ideas to the full linearly distributive case involves adding the structures which we have described above using—the more user friendly but equivalent representation—circuit diagrams.

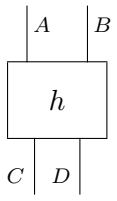
We start by making precise the notion of a **typed circuit**. To build a typed circuit one needs a set of **types**, $\mathcal{T}$, and a set of **components**, $\mathcal{C}$. Each component $f \in \mathcal{C}$ has a signature $\mathrm{sig}(f) = (\alpha, \beta)$, a pair of lists of types, where α is the types of the **input ports** and β the types of the **output ports**.

To obtain a **primitive circuit expression** one attaches to a component two lists of variables. Thus, if $\mathrm{sig}(h) = ([A, B], [C, D])$, then we may write

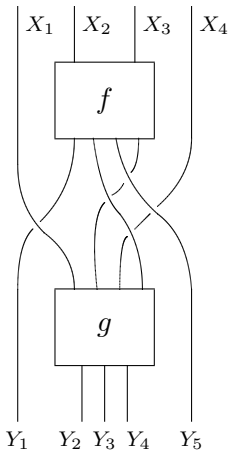
$$h^{x_1,x_2}_{y_1,y_2}$$

the variables in the superscripted list are the **input variables**: each variable must have the correct type for its corresponding port so $x_1 \colon A$ (*viz* x_1 is of type A) and $x_2 \colon B$. The subscripted variable list contains the output variables and they must have types corresponding to the output ports, so $y_1 \colon C$, and $y_2 \colon D$. The variable names in each list must be distinct. The resulting (primitive) circuit expression has a list of input variables $[x_1, x_2]$ and a list of output variables $[y_1, y_2]$.

In the more familiar term logic associated with algebraic theories one does not have output variables. To emphasize that they are something peculiar to this "monoidal" term logic, Lambek referred to them as *covariables*. While we shall not adopt this terminology here, we shall discover that the term does convey their intent.



A primitive circuit presented as a circuit diagram is just a box with a number of (typed) input and output ports. The input wires in this diagram represent the list of typed input variables and the output wires represent the list of typed output variables.



One "plugs" (primitive) circuit expressions together to form new circuit expressions by juxtaposition, just as one attaches two circuit diagrams together:

$$f^{x_2,x_3}_{y_1,z_1,y_5,z_2}; g^{x_1,z_2,x_4,z_1}_{y_2,y_3,y_4}$$

The output variables of the first component which are common to the input variables of the second component become bound in this juxtaposition and indicate how the components are connected. An output variable when it is bound in this juxtaposition is bound to a unique input variable: Lambek would have said that covariables bind to unique variables. To perform a legal juxtaposition the unbound input variables must be distinct and, similarly, the unbound output variables must be distinct. A variable clash occurs when this requirement is violated: one can always rename variables to avoid variable clashes.

When one avoids variable clashes, the juxtaposition operation is associative. Furthermore, when a juxtaposition does not cause any output variable to became bound to any input variables, one can exchange the order of the juxtaposition.

Notice that we have allowed the wires representing the bindings of z_1 and z_2 to "cross", and indeed to access x_1 and x_4 as inputs also requires crossings. Allowing wires to cross in this manner corresponds to having commutativity of the underlying logic. To obtain a noncommutative or *planar* juxtaposition, we would have to properly treat the inputs and outputs as lists of variables instead of viewing them as bags (or multi-sets); this would require that we alter the criteria for juxtaposition (the details are explicitly given in [BCST96]).

A (non-planar) circuit expression C can be **abstracted** by sandwiching it between a non-repeating list of input and output variables. This is written

$$\left\langle C \mid^{x_1,\ldots,x_n}_{y_1,\ldots,y_m} \right\rangle$$

Furthermore one can indicate the types of the input and output wires by the notation:

$$\left\langle C \mid^{x_1:T_1,\ldots,x_n:T_n}_{y_1:T'_1,\ldots,y_m:T'_m} \right\rangle$$

An abstraction must be *closed* in the sense that all the free input variables of C occur in the abstracting input variable list and all the free output variables of C occur in the abstracting output list. Furthermore, any variable in the abstracting input list which is not a free input of C must occur in the abstracting output list and, similarly, any variable in the abstracting output list which is not a free output of C must occur in the abstracting input list.

In particular, we can use this technique of abstracting to isolate a wire (or many wires) as $\left\langle \emptyset \mid^{x:T}_{x:T} \right\rangle$, where $\emptyset$ is the empty circuit and the unit for juxtaposition. This is to be regarded as the "identity map" on the type T: the ability to abstract (and the existence of an empty circuit) are important when we consider how to form categories from circuits.

When a circuit expression is abstracted in this fashion all the wire names become bound; externally an abstraction presents only a list of typed input ports and a list of typed output ports. This permits an abstracted circuit expression to be used as if it were a primitive component. An abstraction used as a component is equivalent to the circuit obtained by removing the abstraction with a substitution of wires *outward* with a renaming of the bound internal wires away from the external wires so as

to avoid variable clashes. To see why the variables of the abstraction are used to substitute the external wires it suffices to consider the use of the "identity" abstraction mentioned above (or indeed any abstraction with "straight-through" wires):

$$\langle\langle \emptyset \mid_{x}^{x} \rangle_{z}^{y} \mid_{z}^{y} \rangle \Longrightarrow \langle \emptyset \mid_{x}^{x} \rangle$$

The operation of removing an abstraction we call abstraction **dissipation**; it is analogous to a β-reduction. The reverse operation is to **coalesce** an abstraction. These operations become particularly important when we consider how one adds identities to the basic circuit identities. In the non-symmetric case, a **planar abstraction** must also preserve the order of the wires.

We may now define the notion of a (non-planar) circuit based on a set of components:

Definition 5.1 (Non-planar circuits)

(i) $\mathcal{C}$**–circuit expressions** *are generated by:*

- *The empty circuit,* $\emptyset$*, is a circuit expression,*
- *If* c_1 *and* c_2 *are circuit expressions which can be juxtapositioned (with no variable clash) then* $c_1; c_2$ *is a circuit expression,*
- *If* $f \in \mathcal{C}$ *is a component with* $\mathrm{sig}(f) = (\alpha, \beta)$ *and* V *is a non-repeating wire list with type* α *and* W *is a non-repeating wire list with type* β *then* f_W^V *is a circuit expression,*
- *If* F *is an abstracted circuit with signature* $\mathrm{sig}(F) = (\alpha, \beta)$ *and* V *is a non-repeating wire list with type* α *and* W *is a non-repeating wire list with type* β *then* F_W^V *is a circuit expression.*

(ii) *A* **circuit** *is an abstracted circuit expression.*

One circuit expression (and by inference circuit) is equivalent to another precisely when one can obtain the second from the first by a series of the following operations:

- Juxtaposition reassociation (with possible bound variable renaming to avoid clashes), $c_1; (c_2; c_3) = (c_1; c_2); c_3$,
- Empty circuit elimination and introduction, $c; \emptyset = c = \emptyset; c$,
- Exchanging non-interacting circuits, $c_1; c_2 = c_2; c_1$,
- Renaming of bound variables,
- Abstraction coalescing and dissipating.

The fact that circuit equivalence under these operations is decidable is immediately obvious when one presents them graphically. Indeed, while it is nice to have a syntax for circuits it is very much more natural and intuitive to simply draw them!

The $\mathcal{C}$–circuits, besides permitting these standard manipulations, can also admit arbitrary additional identities. These take the form of equalities, $c_1 = c_2$, between (closed) abstracted circuits with the same signature. To use such an identity in a circuit, it is necessary to be able to coalesce one of the sides, say c_1 (up to α-conversion) within the circuit. Once this has been done one can replace c_1 with c_2 and dissipate the abstraction. Diagrammatically this corresponds to a surgical operation of cutting out the left-hand side and replacing it with the right-hand side: accordingly such additional identities are often referred to as *rules of surgery*. The circuit reductions and expansions we saw earlier are examples of such rules of surgery.

5.0.2 $\otimes$-circuits: a term logic for monoidal categories

The basic components required to provide a circuit-based term logic for monoidal categories are as follows:

$(\otimes_I)^{A,B}_{A\otimes B}$	$\otimes$–introduction
$(\otimes_E)^{A\otimes B}_{A,B}$	$\otimes$–elimination
$(\top_I)_\top$	unit introduction
$(\top^R_E)^{A,\top}_A$	unit right elimination (thinning)
$(\top^L_E)^{\top,A}_A$	unit left elimination (thinning)

The rules of surgery providing the reduction system for $\otimes$-multi-logic are expressed as follows:

$$\left\langle (\otimes_I)^{x_1,x_2}_z ; (\otimes_E)^z_{y_1,y_2} \; |^{x_1:A,x_2:B}_{y_1:A,y_2:B} \right\rangle \;\Rightarrow\; \left\langle \; |^{x_1:A,x_2:B}_{x_1:A,x_2:B} \right\rangle \tag{1}$$

$$\left\langle (\top_I)_z ; (\top^L_E)^{z,x_1}_{x_2} \; |^{x_1:A}_{x_2:A} \right\rangle \;\Rightarrow\; \left\langle \; |^{x:A}_{x:A} \right\rangle \tag{2}$$

$$\left\langle (\top_I)_z ; (\top^R_E)^{x_1,z}_{x_2} \; |^{x_1:A}_{x_2:A} \right\rangle \;\Rightarrow\; \left\langle \; |^{x:A}_{x:A} \right\rangle \tag{3}$$

The rules of surgery providing the expansion rules for $\otimes$-multi-logic are next. Recall these should be thought of as expressing the type of the wire:

$$\left\langle \; |^{z:A\otimes B}_{z:A\otimes B} \right\rangle \;\Rightarrow\; \left\langle (\otimes_E)^z_{z_1,z_2} ; (\otimes_I)^{z_1,z_2}_z \; |^{z:A\otimes B}_{z:A\otimes B} \right\rangle \tag{4}$$

$$\left\langle \; |^{x:\top}_{x:\top} \right\rangle \;\Rightarrow\; \left\langle (\top_I)_z ; (\top^L_E)^{z,x_1}_{x_2} \; |^{x_1:\top}_{x_2:\top} \right\rangle \tag{5}$$

$$\left\langle \; |^{x:\top}_{x:\top} \right\rangle \;\Rightarrow\; \left\langle (\top_I)_z ; (\top^R_E)^{x_1,z}_{x_2} \; |^{x_1:\top}_{x_2:\top} \right\rangle \tag{6}$$

$$\tag{7}$$

We shall leave as an exercise for the reader the translation of the unit rewirings into this term calculus—graphically they were given in Figure 1. For example, the first one may be written thus:

$$\left\langle (\top^R_E)^{x,z}_x ; (\otimes_I)^{x,y}_w \; |^{x:A,z:\top,y:B}_{w:A\otimes B} \right\rangle = \left\langle (\top^L_E)^{z,y}_y ; (\otimes_I)^{x,y}_w \; |^{x:A,z:\top,y:B}_{w:A\otimes B} \right\rangle$$

As an illustration of the term calculus at work, consider the following example which is the coherence condition for the tensor unit as shown in Figure 3. We shall write the variables $x_1, x_2, \ldots$ as simply $1, 2, \ldots$, numbering the wires, top to bottom, left to right. In this way, the topmost link in the left hand diagram is $(\otimes_E)^1_{2,3}$ (wire 1 comes into the $(\otimes E)$ link, and wires $2, 3$ leave it, 2 on the left, 3 on the right, so that 1 refers to a wire of type $(A \otimes \top) \otimes B$, 2 to a wire of type $A \otimes \top$, and 3 to a wire of type B). Here are the details of the rewriting showing the coalescing, surgery and dissipation

steps:

$$
\begin{aligned}
&\Big\langle (\otimes_E)^1_{2,3}; (\otimes_E)^2_{4,5}; (\otimes_I)^{5,3}_6; (\otimes_I)^{4,6}_7; (\otimes_E)^7_{8,9}; (\otimes_E)^9_{10,11}; (\top^L_E)^{10,11}_{11}; (\otimes_I)^{8,11}_{12} \quad |^1_{12} \Big\rangle \\
&\quad = \quad \Big\langle (\otimes_E)^1_{2,3}; (\otimes_E)^2_{4,5}; (\otimes_I)^{5,3}_6; \Big\langle (\otimes_I)^{4,6}_7; (\otimes_E)^7_{8,9} \; |^{4,6}_{8,9} \Big\rangle^{4,7}_{8,9}; (\otimes_E)^9_{10,11}; (\top^L_E)^{10,11}_{11}; (\otimes_I)^{8,11}_{12} \quad |^1_{12} \Big\rangle \\
&\quad \Rightarrow \quad \Big\langle (\otimes_E)^1_{2,3}; (\otimes_E)^2_{4,5}; (\otimes_I)^{5,3}_6; \Big\langle \quad |^{4,6}_{4,6} \Big\rangle^{4,6}_{8,9}; (\otimes_E)^9_{10,11}; (\top^L_E)^{10,11}_{11}; (\otimes_I)^{8,11}_{12} \quad |^1_{12} \Big\rangle \quad (\otimes\text{-reduction}) \\
&\quad = \quad \Big\langle (\otimes_E)^1_{2,3}; (\otimes_E)^2_{4,5}; (\otimes_I)^{5,3}_6; (\otimes_E)^6_{10,11}; (\top^L_E)^{10,11}_{11}; (\otimes_I)^{4,11}_{12} \quad |^1_{12} \Big\rangle \\
&\quad = \quad \Big\langle (\otimes_E)^1_{2,3}; (\otimes_E)^2_{4,5}; \Big\langle | \; (\otimes_I)^{5,3}_6; (\otimes_E)^6_{10,11} \; |^{5,3}_{10,11} \Big\rangle^{5,3}_{10,11}; (\top^L_E)^{10,11}_{11}; (\otimes_I)^{4,11}_{12} \quad |^1_{12} \Big\rangle \\
&\quad \Rightarrow \quad \Big\langle (\otimes_E)^1_{2,3}; (\otimes_E)^2_{4,5}; \Big\langle \quad |^{5,3}_{5,3} \Big\rangle^{5,3}_{10,11}; (\top^L_E)^{10,11}_{11}; (\otimes_I)^{4,11}_{12} \quad |^1_{12} \Big\rangle \quad (\otimes\text{-reduction}) \\
&\quad = \quad \Big\langle (\otimes_E)^1_{2,3}; (\otimes_E)^2_{4,5}; (\top^L_E)^{5,3}_{11}; (\otimes_I)^{4,11}_{12} \quad |^1_{12} \Big\rangle \\
&\quad = \quad \Big\langle (\otimes_E)^1_{2,3}; (\otimes_E)^2_{4,5} \Big\langle (\top^L_E)^{z,y}_y; (\otimes_I)^{x,y}_w \; |^{x,z:\top,y}_w \Big\rangle^{4,5,3}_{12} \quad |^1_{12} \Big\rangle \quad (\text{coalescing}) \\
&\quad = \quad \Big\langle (\otimes_E)^1_{2,3}; (\otimes_E)^2_{4,5} \Big\langle (\top^R_E)^{x,z}_x; (\otimes_I)^{x,y}_w \; |^{x:A,z:\top,y:B}_{w:A\otimes B} \Big\rangle^{4,5,3}_{12} \quad |^1_{12} \Big\rangle \quad (\text{surgery: tensor unit axiom}) \\
&\quad = \quad \big\langle (\otimes_E)^1_{2,y}; (\otimes_E)^2_{x,z} (\top^R_E)^{x,z}_x; (\otimes_I)^{x,y}_w \quad |^1_w \big\rangle \quad (\text{dissipation}) \\
&\quad = \quad \Big\langle (\otimes_E)^1_{2,3}; (\otimes_E)^2_{4,5} (\top^R_E)^{4,5}_4; (\otimes_I)^{4,3}_6 \quad |^1_6 \Big\rangle \quad (\text{renaming}).
\end{aligned}
$$

This amply illustrates why it is so attractive to work with the diagrammatic representation! However, it is important to know that under the hood of circuit diagrams there is a fully formal (if somewhat verbose) notion of circuits which, for example, could be implemented on a computer.

6 Functor boxes

The use of circuit diagrams—and similar graphical tools using similar but different conventions—is very widespread, and includes applications to systems developed for handling quantum computing (dagger categories) and for fixpoints and feedback (traced monoidal categories). They are even a useful device in understanding the categorical proof theory of classical logic: it is well-known that the usual Lambek-style approach collapses to postal proof theory, so one cannot distinguish between proofs of the same sequents. However, in [FP07, LS05, LS06] (and others) for example, interesting non-posetal proof theory for classical logic is obtained by construction on top of the proof theoretic substrate provided by linearly distributive categories or $*$-autonomous categories.

A somewhat different application of circuits was introduced in [CS97c], which may be applied to obtain a variety of logics (including a simple linear modal logic [BCS02]), which described circuits for structured functors. The basic idea is similar to the proof boxes of J-Y Girard [G87] for the exponentials **!** and **?**, as described in [BCS96]. But for more general functors, a slightly different approach was needed, which we shall sketch here. Full details are available in [CS97c].

The first question to address is how to handle functors, indeed, "why do functors at all?" (*i.e.* "why boxes?"!). We shall see an example at the end of this section: modal logic. The first use of

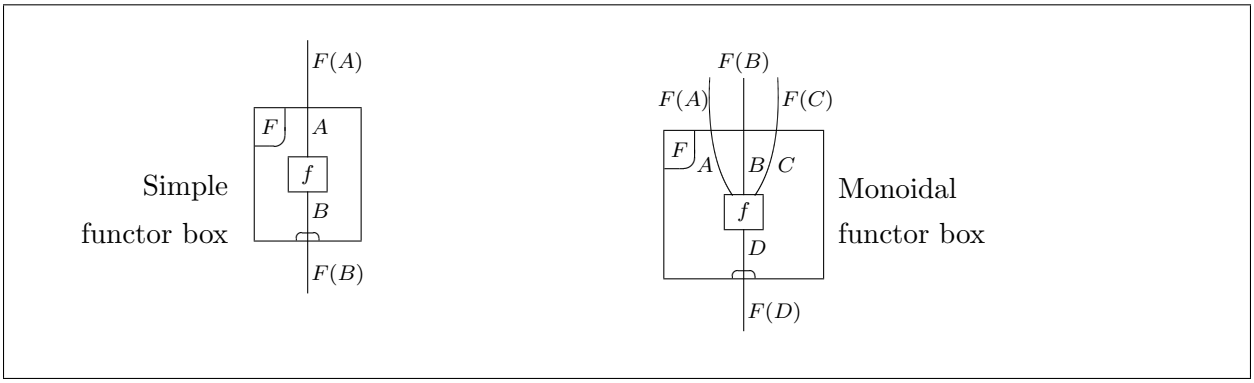


Figure 6: Simple and monoidal functor boxes

boxes was Girard's approach in [G87] to the exponentials, which were called modalities from the beginning. They were necessary to be able to interpret intuitionistic logic in linear logic, and in fact were really the reason for his initial development of linear logic. More traditional modal logic would also seem to require functors for $\Box$ and $\Diamond$ (necessity and possibility). But we must handle these at the level of derivations (as well as formulas), so in effect we need to be able to "apply" the modalities to morphisms as well as to formulas. And the simplest way (or so it seems!) is to simply take a subgraph corresponding to a derivation, and replace it with one corresponding to the image of that derivation under the modality. And that is easily handled with the boxes we shall describe below. Interestingly, this notion was independently discovered by Paul-André Melliès [M06].

We start with an ordinary functor $F\colon \mathbf{C} \to \mathbf{D}$; given a morphism $f\colon A \to B$ in $\mathbf{C}$, represented as a component with input wire of type A and output wire of type B, the corresponding morphism $F(f)\colon F(A) \to F(B)$ in $\mathbf{D}$ is represented by simply "boxing" the component, as shown in Figure 6.

Note that the box bears a label with the name of the functor. These functor boxes have one input and one output; if the component f is a poly-map, then it is necessary to tensor the inputs and par the outputs to obtain a one-in-one out map before the functor box can be applied. We shall relax this condition soon in discussing monoidal and linear functors. The half oval through which the wire leaves the box is called the "principle port"; this is not really essential here, but we include it for comparison with the structured boxes that will be described next: at that point its role will become clear. Notice also the typing changes the box imposes on a wire as it passes into or out of a box.

There are two obvious rewrites to express functoriality: an "expansion" which takes an identity wire of type $F(A)$ and replaces it with an identity wire of type A which is then "boxed", and a "reduction" which "merges" two functor boxes one of which directly "feeds" into the next (box "eats" box).

Now we consider the situation where there is some structure on both categories and functor: suppose first that the categories are monoidal categories, and the functor is also monoidal, meaning that preservation of the tensor is lax: the functor F is monoidal if there are natural transformations $m_\otimes : F(A) \otimes F(B) \to F(A \otimes B)$ and $m_\top : \top \to F(\top)$ satisfying the equations

$$\begin{aligned} u^L_\otimes = m_\top \otimes 1; m_\otimes; F(u^L_\otimes) \quad &: \quad \top \otimes F(A) \to F(A) \\ a_\otimes; 1 \otimes m_\otimes; m_\otimes = m_\otimes \otimes 1; m_\otimes; F(a_\otimes) & \\ &: \quad (F(A) \otimes F(B)) \otimes F(C) \to F(A \otimes (B \otimes C)) \end{aligned}$$

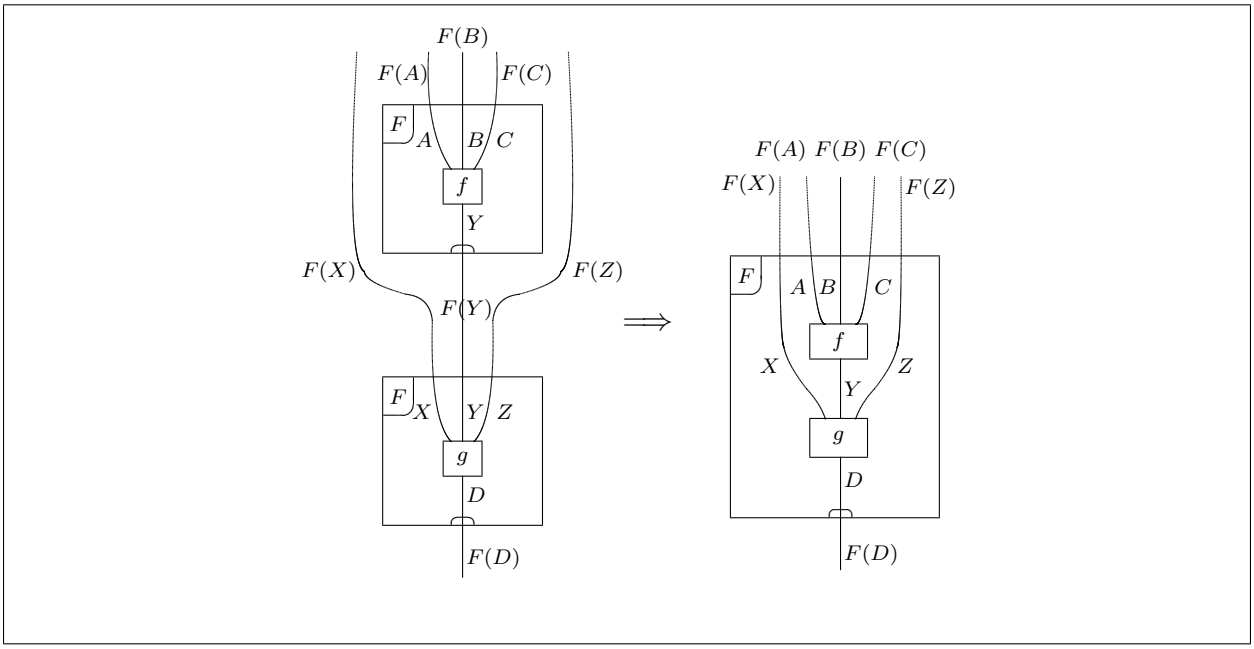


Figure 7: Box-eats-box rule

and in the symmetric case, the next equation as well:

$$m_\otimes; F(c_\otimes) = c_\otimes; m_\otimes \quad : \quad F(A) \otimes F(B) \to F(B \otimes A)$$

We will soon also want the dual notion (for the dual par $\oplus$): a functor G is comonoidal if there are natural transformations $n_\oplus : G(A \oplus B) \to G(A) \oplus G(B)$ and $n_\bot : G(\bot) \to \bot$ satisfying equations dual to those above.

To capture the effect in circuits of requiring a functor to be monoidal, we modify the effect of the "functor box", as shown in Figure 6.

Note that for F monoidal, we may relax the supposition that the boxed subgraph is "one-in-one-out" to allow multi-arrows which have many input wires (but still just one output wire), as suggested in the figure above.

One might expect that we would have to add components representing the two natural transformations $m_\otimes, m_\top$ that are necessary for F to be monoidal. However, it is an easy exercise to show that these can be induced by the formation rule for monoidal functor boxes: $m_\otimes$ is the case where f is the $(\otimes\ I)$ node (two inputs, A, B and one output $A \otimes B$), and $m_\top$ is the case where f is the $(\top\ I)$ node (no inputs and one output $\top$).

There are reduction and expansion rewrites for these boxes (and one for handling "twists" permitted by symmetry). The necessary reduction rewrite is shown in Figure 7 (we refer to this saying one box "eats" the other). And the "expansion" rule and the "twist" rule are shown in Figure 8

We have already indicated what the nets are for $m_\otimes, m_\top$. It is a fairly straightforward exercise to show that the equations are consequences of the net rewrites given above, and that the rewrites correspond to commutative diagrams, if F is monoidal. (An exercise we leave for the reader(!)—but recall that the details may be found in [CS97c].) As an example, Figure 9 shows that the equation dealing with "reassociation" is true for any F whose functor boxes satisfy the circuit rewrites we have given so far. So this circuit syntax is indeed sound and complete for monoidal functors. For

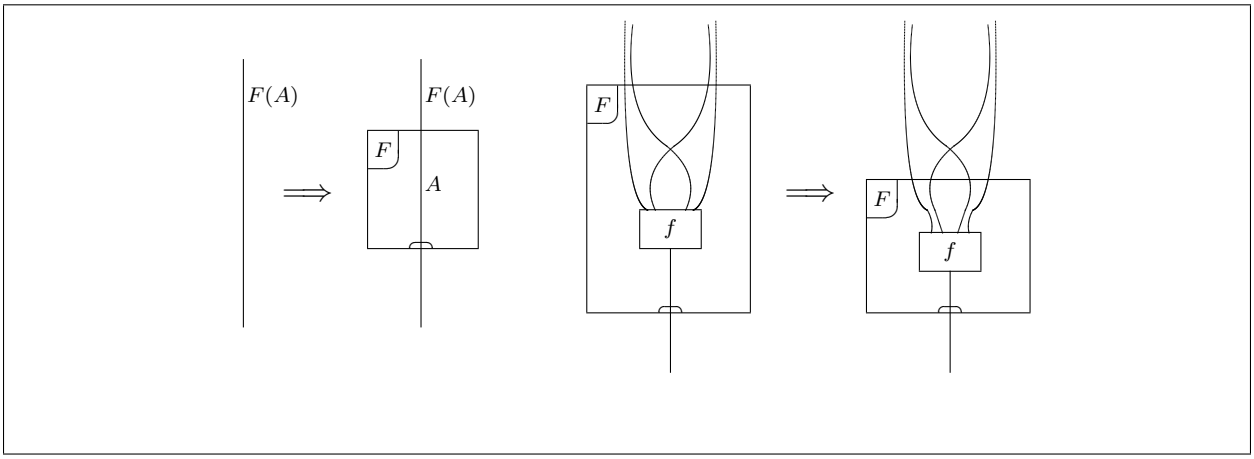


Figure 8: Expansion and twist rules

comonoidal functors, we just use a dual syntax, with the corresponding rewrites. Note then that for comonoidal functors, the principle port will be at the top of the box (this is the role of the principle port, to distinguish monoidal functors from comonoidal ones).

6.1 Linear functors

To handle linear logic, it turns out that the suitable notion of functor is what we call a "linear functor": this is really a pair of functors related by a shadow of duality. This duality becomes explicit in the presence of negation (*i.e.* when the category is $*$-autonomous): then the pair of functors are de Morgan duals: all this is discussed in more detail in [CS97c].

So, a linear functor $F\colon \mathbf{C} \to \mathbf{D}$ between linearly distributive categories $\mathbf{C}, \mathbf{D}$, consists of:

1. a pair of functors $F_\otimes, F_\oplus : \mathbf{X} \to \mathbf{Y}$ so that $F_\otimes$ is monoidal with respect to $\otimes$, and $F_\oplus$ is comonoidal with respect to $\oplus$,
2. natural transformations (called "linear strengths")

$$\begin{array}{rcl} \nu^R_\otimes & : & F_\otimes(A \oplus B) \to F_\oplus(A) \oplus F_\otimes(B) \\ \nu^L_\otimes & : & F_\otimes(A \oplus B) \to F_\otimes(A) \oplus F_\oplus(B) \\ \nu^R_\oplus & : & F_\otimes(A) \otimes F_\oplus(B) \to F_\oplus(A \otimes B) \\ \nu^L_\oplus & : & F_\oplus(A) \otimes F_\otimes(B) \to F_\oplus(A \otimes B) \end{array}$$

satisfying coherence conditions corresponding to the requirements that the linear strengths are indeed strengths, and that the various transformations are compatible with each other. (These are listed explicitly in [CS97c].) A representative sample is given here—the rest are generated by the obvious dualities.

$$\begin{array}{rcl} \nu^R_\otimes; n_\perp \oplus 1; u^L_\oplus & = & F_\otimes(u^L_\oplus) \\ F_\otimes(a_\oplus); \nu^R_\otimes; 1 \otimes \nu^R_\otimes & = & \nu^R_\otimes; n_\oplus \oplus 1; a_\oplus \\ F_\otimes(a_\oplus); \nu^R_\otimes; 1 \otimes \nu^L_\otimes & = & \nu^L_\otimes; \nu^R_\otimes \oplus 1; a_\oplus \\ 1 \otimes \nu^R_\otimes; \delta^L_L; \nu^R_\oplus \oplus 1 & = & m_\otimes; F_\otimes(\delta^L_L); \nu^R_\otimes \\ 1 \otimes \nu^L_\otimes; \delta^L_L; m_\otimes \oplus 1 & = & m_\otimes; F_\otimes(\delta^L_L); \nu^L_\otimes \end{array}$$

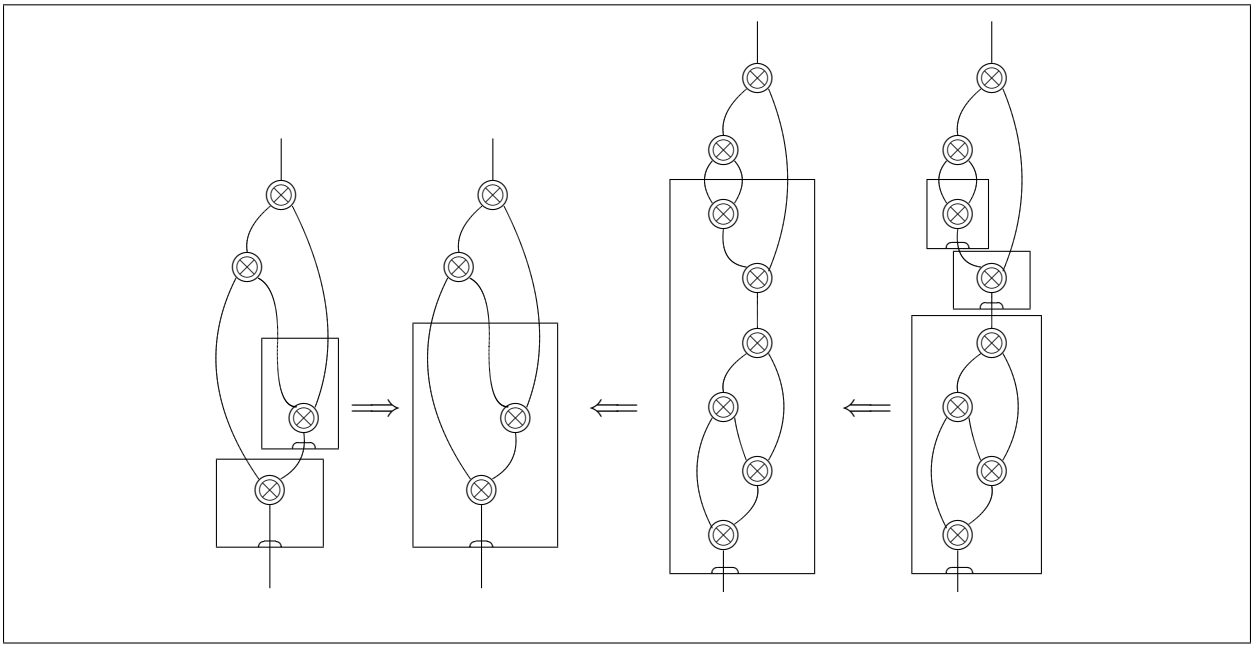


Figure 9: Functor boxes are monoidal—reassociation

In [CS97c] the definition of linear transformations, which are necessary to describe the 2-categorical structure of linear logic, is also given; but we shall not pursue that further here.

Now we extend the syntax of functor boxes to linear functors; in fact all that is necessary is to generalize the monoidal boxes to allow the boxed circuit to have arbitrarily many inputs and outputs. So for the monoidal component $F_\otimes$ of a linear functor F, the functor boxes will have the formation rule shown in Figure 10, and the comonoidal component will have the dual rule (just turn the page upside down). Please note the typing of this formation rule carefully: at the top of the box, the functor applied is the functor $F_\otimes$ associated with the box, but at the bottom, only the wire that leaves through the principal port gets an $F_\otimes$ attached to it, the other wires get the comonoidal $F_\oplus$ attached to them. (The dual situation applies for the $F_\oplus$ boxes.) This is the role of the principal port in our notation (and is similar to the notation used in [BCS96] for the "exponential" or "modal" operators **!** and **?**). There may be only one principal port, though there may be arbitrarily many other ("auxiliary") ports.

It is then quite easy to represent the $\nu_\otimes^R$ map as a boxed $(\oplus\ E)$ node—the right output wire of the node passes through the principal port. The three other linear strengths are given similarly: $\nu_\otimes^L$ is the $(\oplus\ E)$ node boxed with a $F_\otimes$ box, the left output wire passing through the principal port. The two $\nu_\oplus$ maps are given by the $(\otimes\ I)$ node boxed by the $F_\oplus$ box, with either the right or the left input ports being the principal port, as appropriate.

Associated with these box formation rules are several rewrites. The expansion rewrite remains as before, but the reduction rewrite must be generalized to account for the more general f; this is done in the obvious fashion. Similarly, in the symmetric case we generalize the rewrites that move a "twist" outside a box; in fact, in the symmetric case it is convenient to regard the order of inputs/outputs as irrelevant, so that these rewrites are in fact equalities of circuits. In addition, we must account for the interaction between $F_\otimes$ and $F_\oplus$ boxes, which gives a series of rewrites that allow one box to "eat" another whenever a non-principal wire of one type of box becomes the

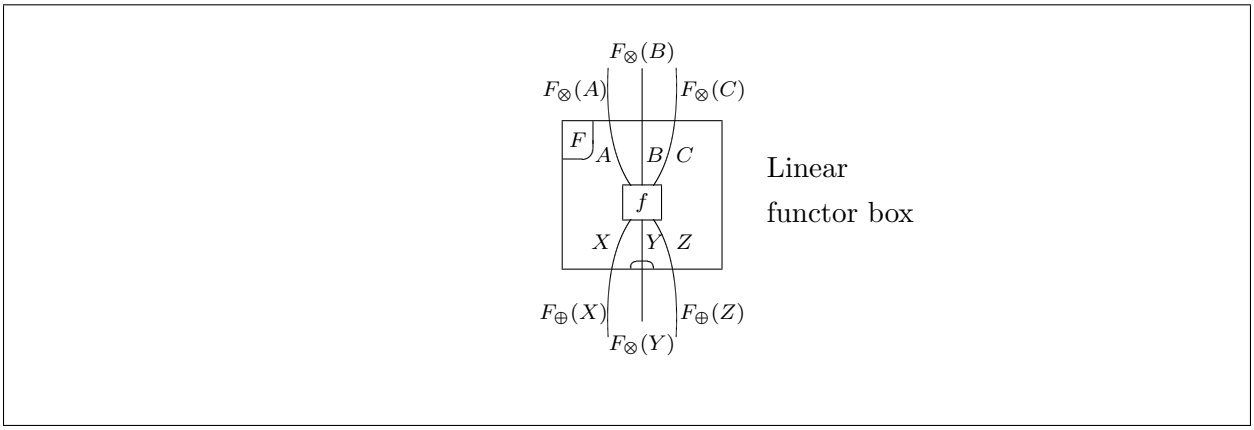


Figure 10: Linear functor box

principal wire of the dual type. We give an example of this in Table 1, along with the other rules mentioned in this paragraph. The reader may generate the dual rules. Note that in this Table we have illustrated circuits with crossings of wires; in the noncommutative case, such crossings must not occur, so some wires must be absent from these rewrites. The rewrites dealing with pulling a "twist" out of a box are only relevant in the symmetric case of course. We have illustrated one, where the "twisted" wires are inputs; it is also possible that the "twisted" wires are outputs, and one may (or may not) be the wire through the principle port.

These rules are sound and complete; we must verify that any $F = (F_\otimes, F_\oplus)$ which allows such a calculus is indeed a linear functor, and conversely, that any linear functor allows such rewrites [CS97c]. More interestingly, however, is what can be done with such functors (and their circuits). The key fact is that all the basic structure of linear logic (with the exception of negation) can be described in terms of linear functors: in a natural manner, tensor and par together form a linear functor, as do Cartesian product and coproduct. And not surprisingly, so do the exponentials **!** and **?**. If negation is added to the mix (so the category is $*$-autonomous, not merely linearly distributive), then each pair consists of de Morgan duals, and in that context, the pair really just amounts to the monoidal functor in the pair (the comonoidal functor just being its de Morgan dual).

There is a natural logic associated with a linear functor. A special case is illuminating: consider a linear functor $F\colon \mathbf{X} \to \mathbf{X}$; write $\Box$ for $F_\otimes$ and $\diamondsuit$ for $F_\oplus$. We shall not give a complete description of the logic one obtains from this (see [BCS02]), but here are some highlights:

$$\begin{aligned} \nu^L_\otimes\colon \Box\,(A \oplus B) &\to \Box A \oplus \diamondsuit B \\ m_\otimes\colon \Box\, A \otimes \Box B &\to \Box(A \otimes B) \end{aligned}$$

In a classical setting, these would be equivalent to

$$\begin{aligned} \Box(A \Rightarrow B) &\to (\Box A \Rightarrow \Box B) \\ \Box A \wedge \Box B &\to \Box(A \wedge B) \end{aligned}$$

the first being "normality" of the logic, and the second being one half (the linear half!) of the standard isomorphism

$$\Box A \wedge \Box B \longleftrightarrow \Box(A \wedge B)$$

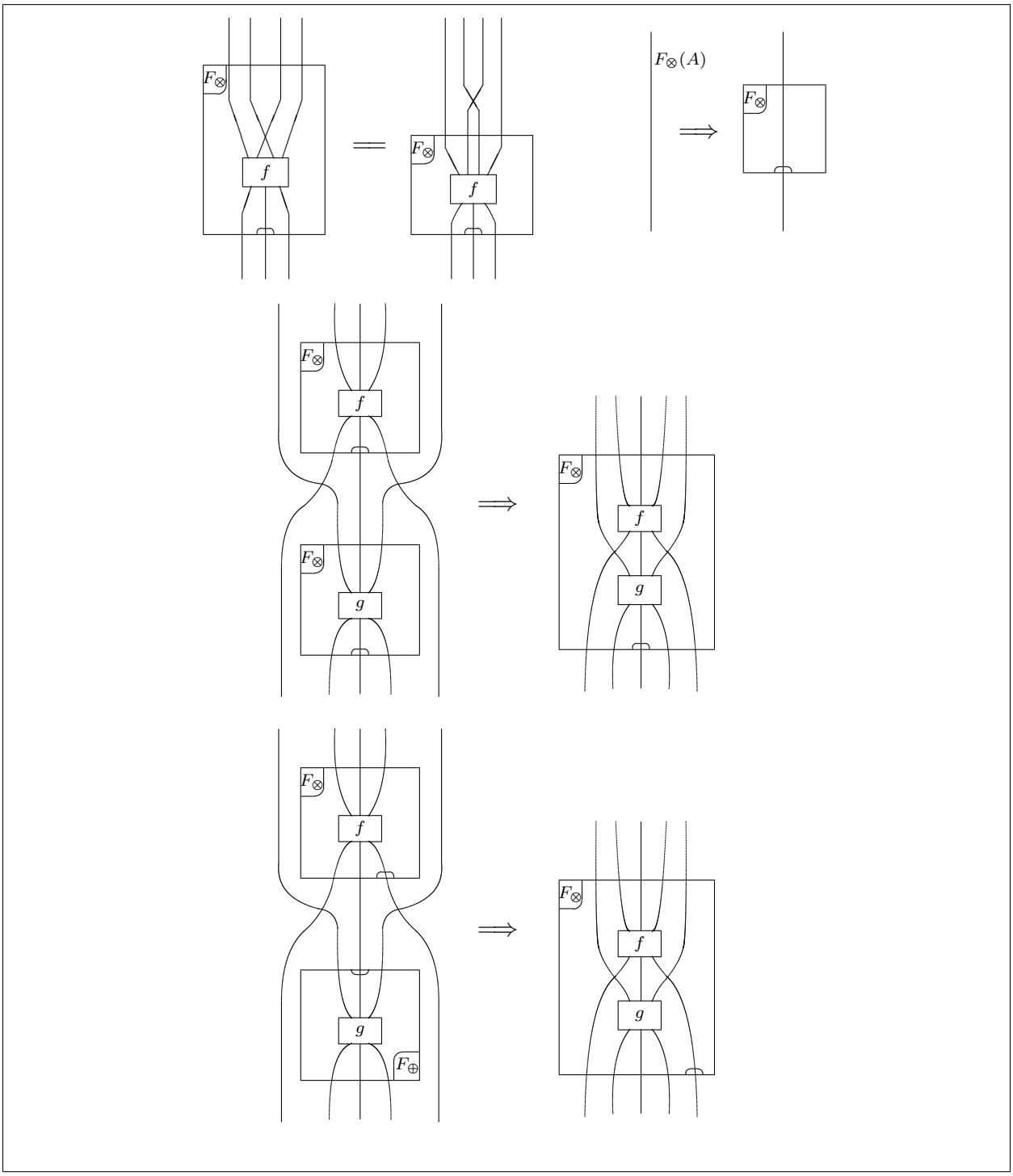


Table 1: Some reduction and expansion rewrites for functor boxes

The following rule (and its dual cousins) holds in basic linear modal logic:

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

This is a familiar rule in many modal situations, for example it occurs in the process calculus of Hennessy and Milner [HM85].

References

[BCS96] R.F. Blute, J.R.B. Cockett, and R.A.G. Seely "**!** and **?**: Storage as tensorial strength", Mathematical Structures in Computer Science **6** (1996), 313–351.

[BCST96] R.F. Blute, J.R.B. Cockett, R.A.G. Seely, and T.H. Trimble "Natural deduction and coherence for weakly distributive categories", Journal of Pure and Applied Algebra **113** (1996), 229–296.

[BCS02] R.F. Blute, J.R.B. Cockett, and R.A.G. Seely "The logic of linear functors", Mathematical Structures in Computer Science **12** (2002), 513–539.

[CS97a] J.R.B. Cockett and R.A.G. Seely "Weakly distributive categories", Journal of Pure and Applied Algebra **114** (1997), 133–173.
(Updated version available on <http://www.math.mcgill.ca/rags>.)

[CS97b] J.R.B. Cockett and R.A.G. Seely "Proof theory for full intuitionistic linear logic, bilinear logic, and MIX categories", Theory and Applications of Categories **3** (1997), 85–131.

[CS97c] J.R.B. Cockett and R.A.G. Seely "Linearly distributive functors", Journal of Pure and Applied Algebra **143** (1999), 155–203.

[CKS03] J.R.B. Cockett, J. Koslowski, and R.A.G. Seely "Morphisms and modules for poly-bicategories", Theory and Applications of Categories **11** (2003), 15–74.

[CKS00] J.R.B. Cockett, J. Koslowski, and R.A.G. Seely "Introduction to linear bicategories", Mathematical Structures in Computer Science **10** (2000), 165–203.

[DR89] Danos, V. and L. Regnier "The structure of multiplicatives", Archive for Mathematical Logic **28** (1989), 181–203.

[FP07] Fuhrmann, C. and Pym, D. "On categorical models of classical logic and the geometry of interaction", Mathematical Structures in Computer Science **17** (2007), 957-1027.

[G87] Girard, J.-Y. "Linear logic", Theoretical Computer Science **50** (1987), 1–102.

[HM85] Hennessy, M. and R. Milner "Algebraic laws for nondeterminism and concurrency", Journal of the ACM **32** (1985), 137–161.

[HH14] Heijltjes, W, and R. Houston "No proof nets for MLL with units: proof equivalence in MLL is PSPACE-complete", CSL-LICS '14, Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) Article No. 50 (2014).

[H00] Hermida, C. "Representable multicategories". Advances in Mathematics (2000), 164–225.

[JS91] Joyal, A. and R. Street "The geometry of tensor calculus I", Advances in Mathematics **88** (1991), 55–112.

[LS05] Lamarche, F, and L Straßburger "Constructing free Boolean categories", in LICS Proceedings, IEEE Press, 2005.

[LS06] Lamarche, F, and L Straßburger "From proof nets to the free $*$-autonomous category", Logical Methods in Computer Science **2**(4:3), 2006.

[L69] Lambek, J. "Deductive systems and categories II", Lecture Notes in Mathematics **87** (Springer-Verlag, Berlin, Heidelberg, New York, 1969).

[L89] Lambek, J. "Multicategories revisited", in J. Gray and A. Scedrov (eds.), *Categories in Computer Science and Logic*, Contemporary Mathematics **92** (Am. Math. Soc. 1989), 217–237.

[M06] Melliès, P.-A. "Functorial Boxes in String Diagrams", Lecture Notes in Computer Science **4207**, 1–30 (Springer-Verlag, Berlin, Heidelberg, New York, 2006)

[P65] Prawitz, D. *Natural Deduction*, Almqvist and Wiksell, Uppsala (1965)

[P71] Prawitz, D. "Ideas and results in proof theory", in J. Fenstad (ed.), *Proceedings of the Second Scandinavian Logic Symposium*, Amsterdam: North-Holland (1971), 235308.

[Se89] Seely, R.A.G. "Linear logic, $*$-autonomous categories and cofree coalgebras", in J. Gray and A. Scedrov (eds.), *Categories in Computer Science and Logic*, Contemporary Mathematics **92** (Am. Math. Soc. 1989), 371–382.

[S11] Selinger, P. "A survey of graphical languages for monoidal categories", in B. Coecke (ed.), *New Structures for Physics*, Lecture Notes in Physics **813**, 289355, Springer, 2011.

[Sch99] Schneck, R.R. "Proof theory: Natural deduction and coherence for non-symmetric linearly distributive categories", Theory and Applications of Categories **6** (1999) 105–146.