Double categories:
a modular model of multiplicative linear logic

Paul-André Melliès

Équipe Preuves, Programmes, Systèmes

CNRS, Université Paris 7

<melllies@pps.jussieu.fr>

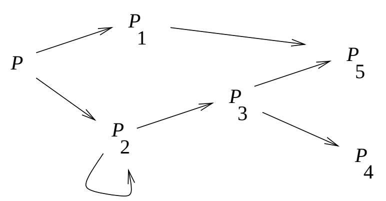
Received

We construct a double category $\mathcal{D}$ of proof-nets in multiplicative linear logic (MLL). Its horizontal arrows are MLL modules (subnets of well-formed nets), its vertical arrows model side-effects, and its double cells interpret the cut-elimination procedure. The categorical model is *modular* in the following sense: every computation of a composite module $(\pi_1; \pi_2)$ factors out as the separate and interacting computations of the two subcomponents π_1 and π_2. This enables to *trace* MLL modules in the course of cut-elimination, and analyze their behaviour in time.

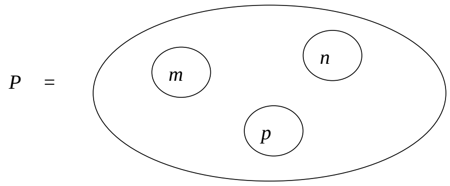
1. Motivations

Two theoretical traditions have guided so far the analysis of programming languages and computation: on one hand *syntax* and its focus on rewriting, on the other hand *semantics* and its interest in meanings. Our purpose here is to relate them, by modularity.

Syntax, or how a term evolves in the course of computation. A rewriting system Σ defines a transition graph whose vertices P and edges $P \to Q$ are the terms and reduction steps of Σ. Rewriting theory is concerned with the properties of that graph, and the dynamics of syntax it reflects. The main tool to analyze that dynamics is to *trace* reduction steps in the course of computation, see (Lévy 1978; Huet Lévy 1979; Klop 1992; Gonthier Lévy Melliès 1992) for a theory of residuals and its applications.



Semantics, or what a program means to its environment. Every program P (=term) is the combination of several procedures m, n or p (=subterms) whose interaction is what one generally calls "the computation of P". Denotational semantics assigns to each procedure m and program P a *meaning* retaining the operation it performs, or the function it computes — without the unnecessary details of implementation. The theory ensures that two procedures m and m' with the same meaning may be safely interchanged in a program P.



Syntax as semantics, and conversely! If traditionally, the meaning of a program indicates *what* function it computes, recent developments around geometry of interaction and game semantics add to this "static" meaning the "dynamic" information of *how* the program interacts with its environment. In doing so, semantics reformulates in its own language (cartesian or monoidal closed categories) problems traditionally seen as syntactical, like abstract machines and complexity. In this article, we attack the converse problem of explaining *semantics* as an effect of *syntax*, by extracting the *meaning* of a program from its *dynamics*. Our analysis† will be guided by the following thesis:

" The meaning of a module should reflect the history of its symbolic interactions with the environment " (1)

In this article, we deliver a simple methodology to extract the meaning of a module from its interactive behaviour. We explain in what sense this relates to tracing modules in the course of computation.

Spy vs. spy. Let us choose a *diplomatic* illustration of this idea. Suppose that you — an under-cover spy — sit in a café with two members of your secret organization, p and q, and wonder whether p is *loyal* or *traitor* to your cause. To solve this vital question, you will interact with p and q with a motivated interest to your friend p. Of course, the only way to obtain valuable information on p is to trace him in the course of discussion, and study the sequence of his reactions. The task would be hopeless if p disappears, or worse, if p and q interact so well as to mix together, without a secure way for you to distinguish one from the other (a plausible nightmare). Let us say that the same constraint holds for a term M (= the global café conversation) and its modules (= the friends p or q): to derive the meaning of a module p from its syntactical behaviour requires to trace p in the course of the computation of M.

Let us trace modules! Tracing subterms is a tradition in rewriting theory. In the λ-calculus for instance, letting $C[-]$ denote a λ-context, Lévy explains in (Lévy 1978) how a β-redex

$$C[(\lambda x.M)P] \to C[M[P/x]]$$

† From now on, we use the word *module* instead of *procedure*, to conform with the fashion of linear logic (Girard 1987b).

may be traced in the course of alternative computations of $C[(\lambda x.M)P]$. Unfortunately, nobody knows exactly (today) how to extend the technique to trace properly a (higher-order) λ-term P through the computations of $C[P]$ — see (Lévy 1978; Melliès 1996) for two partial solutions. In this article we forget the algebraic attempts of (Lévy 1978; Melliès 1996) and shift to a more elementary description of rewriting: enriched categories, where we partly realize our tracing ambitions.

A syntactic category $\mathfrak{L}$ for the simply typed λ-calculus may be defined along the lines of (Lambek Scott 1986). Simple types (hereafter denoted A, B or C) are constructed from the ground type ι and connective $A \to B$. A context is a sequence of the form

$$x_0 : A_0, \ldots, x_{m-1} : A_{m-1}$$

where the x_i's are variables and the A_i's are simple types. The typing judgement $\vdash$ between contexts, λ-terms [‡] and simple types is the smallest relation which verifies:

$$(\textsc{Var})\ \Gamma, x : A \vdash x : A \qquad (\textsc{Weak})\ \frac{\Gamma \vdash M : B}{\Gamma, x : A \vdash M : B}\quad x \text{ not free in } M$$

$$(\textsc{Lambda})\ \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \qquad (\textsc{App})\ \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

The category $\mathfrak{L}$ is defined with contexts as objects, and morphisms

$$\Delta = (x_0 : A_0, \ldots, x_{m-1} : A_{m-1}) \ \longrightarrow\ \Gamma = (y_0 : B_0, \ldots, y_{n-1} : B_{n-1})$$

the sequences $(M_0, \cdots, M_{n-1})$ of λ-terms M_i such that $\Delta \vdash M_i : B_i$ for every $0 \leq i \leq n-1$. Composition in $\mathfrak{L}$ is defined as λ-term substitution. For instance:

$$\Delta \xrightarrow{\ \lambda x.M\ } (y : A \to B) \xrightarrow{\ yP\ } z : B \tag{2}$$

compose as:

$$\Delta \xrightarrow{\ (\lambda x.M)P\ } z : B \tag{3}$$

2-categories allow to enrich syntactic categories with a notion of "reduction" — see (Seely 1987; Rydeheard Stell 1987; Power 1989; Jay 1992; Ghani 1995). A 2-category $\mathcal{C}_2$ is a category whose hom-sets $\mathcal{C}_2(\Delta, \Gamma)$ are categories themselves, with suitable coherence properties. A morphism $\alpha : M \Rightarrow N$ in the hom-category $\mathcal{C}_2(\Delta, \Gamma)$ is called a *cell* and is usually represented as follows:

$$\begin{tikzcd} \Delta \arrow[r, bend left, "M"] \arrow[r, bend right, "N"'] \arrow[r, phantom, "\Downarrow\alpha"] & \Gamma \end{tikzcd}$$

The "β-reduction" 2-category $\mathfrak{L}_2$ associated to the λ-calculus has the same objects and morphisms as $\mathfrak{L}$, and cells $\alpha : M \Rightarrow N$ the sequences of β-reductions from M to N (modulo so-called Lévy permutation equivalence). This means for instance that in $\mathfrak{L}_2$

‡ identified up to α-conversion: $\lambda x.x = \lambda y.y$.

there exists a cell α from the morphism (3) to $M[P/x]$:

$$\begin{tikzcd}
& (y : A \to B) \arrow[dr, "yP"] \arrow[d, phantom, "\Downarrow\alpha"] & \\
\Delta \arrow[ur, "\lambda x.M"] \arrow[rr, "{M[P/x]}"'] & & z : B
\end{tikzcd} \tag{4}$$

The 2-category $\mathcal{L}_2$ is not satisfactory for term tracing. By diagrams (2) and (3), the λ-term $(\lambda x.M)P$ factors out as a witness for $(\lambda x.M)$ and a witness P. After one β-reduction (4), the factorization of $M[P/x]$ is still possible, but reversed, as follows:

$$\Delta \xrightarrow{\mathrm{id}_\Delta, P} \Delta, x : A \xrightarrow{M} z : B \tag{5}$$

where, to simplify, we assume that the variable x occurs exactly once in M. In that sense, the λ-term P is *visible* in the two λ-terms $(\lambda x.M)P$ and $M[P/x]$. Unfortunately, this "visibility as factorization" of P disappears after further β-reductions. In fact, a reduction cell γ

$$\begin{tikzcd}
\Delta \arrow[r, bend left, "(\lambda x.M)P"] \arrow[r, bend right, "Q"'] \arrow[r, phantom, "\Downarrow\gamma"] & z : B
\end{tikzcd}$$

entangles P and $(\lambda x.M)$ so well generally that no witness for $(\lambda x.M)$ or P factors out from the morphism Q.

However, if we want to trace the λ-terms $(\lambda x.M)$ and P in $(\lambda x.M)P$, we should be able to factor them out from Q. More conceptually, this factorization should derive from the decomposition of the computation γ itself into interactions between $(\lambda x.M)$ and P. In categorical terms, this means that the cell γ should factor out as a cell γ_1 from $(\lambda x.M)$ and a cell γ_2 from P. If we try to apply the principle on the cell (4) and factor α into two cells α_1 and α_2, we obtain a diagram like:

$$\begin{tikzcd}
& (y : A \to B) \arrow[dr, "yP"] \arrow[dd, dotted, "\alpha"] & \\
\Delta \arrow[ur, "\lambda x.M"] \arrow[dr, "M"'] \arrow[r, phantom, "\alpha_1\Downarrow"] & {} & z : B \arrow[l, phantom, "\Downarrow\alpha_2"'] \\
& (z' : B, x : A^\perp) \arrow[ur, "{z', x := P}"'] &
\end{tikzcd} \tag{6}$$

Here the $x := P$ should be interpreted as a "reversed" substitution whose *right* composition proceeds what the *left* composition proceeds in (5): the substitution of P for the variable x in M. The type $A^\perp$ which is the "input" dual of A indicates this opposite orientation in substitutions. Finally, the λ-term z' connects $M[P/x]$ to z.

Observe that the decomposition of α into α_1 and α_2 is not possible in the 2-category $\mathcal{L}_2$ because α_1 and α_2 have "side-effects" on their interface, expressed by the vertical arrow

$$(y : A \to B) \overset{a}{\cdots\cdots\cdots\rightarrow} (x : A^{\perp}, z' : C)$$

This vertical arrow a stands for the synchronized emission of an application link by α_2 and of a (λx) link by α_1.

The idea to decompose computations γ as separate interactions γ_1 and γ_2 in order to trace subterms and express modularity properties was proposed independently by the author in (Melliès 1994) and by Gadducci and Montanari in (Gadducci Montanari 1996). The advantage of (Gadducci Montanari 1996) over (Melliès 1994) is to express the decomposition inside a double category. See also the later work of (Bruni Montanari 1999). We follow this line here and introduce double categories.

A double category is an internal category of the category of all categories CAT, see (Jacobs 1991). More explicitly, a double category $\mathcal{D}$ consists of a collection of objects $\Delta, \Gamma, ...$, horizontal arrows $M, N, P, ...$, vertical arrows $a, b, c, ...$ and cells $\alpha, \beta, \gamma, ...$ such that:

— objects and horizontal arrows form a category $\mathcal{D}^h$, called the *horizontal category*, with composition $M; N$ and identities id_{Δ},

— objects and vertical arrows form a category $\mathcal{D}^v$, called the *vertical category*, with composition $a \cdot b$ and identities id^{Δ},

— any cell α is assigned an horizontal source a and an horizontal target b, notation: $\alpha : a \Rightarrow_h b$,

— any cell α is assigned a vertical source M and vertical target N, notation: $\alpha : M \Rightarrow_v N$,

Horizontal, vertical, source and target arrows of a cell α must be "compatible" in the following graphical sense:

$$\begin{tikzcd} \Delta \arrow[r, "M"] \arrow[d, dotted, "a"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "b"] \\ \Delta' \arrow[r, "N"'] & \Gamma' \end{tikzcd} \quad \text{when} \quad \begin{array}{c} \alpha : a \Rightarrow_h b \\ \alpha : M \Rightarrow_v N \end{array}$$

In addition, cells may be composed horizontally as $(\alpha *_h \beta)$ and vertically as $(\gamma *_v \delta)$. Typically, two cells $\alpha : a \Rightarrow_h b$ and $\beta : b \Rightarrow_h c$ are composed horizontally as $\alpha *_h \beta : a \Rightarrow_h c$ in the following way:

$$\begin{tikzcd} \Delta \arrow[r, "M"] \arrow[d, dotted, "a"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[r, "N"] \arrow[d, dotted, "b"] \arrow[dr, phantom, "\beta"] & \Xi \arrow[d, dotted, "c"] \\ \Delta' \arrow[r, "M'"'] & \Gamma' \arrow[r, "N'"'] & \Xi' \end{tikzcd} = \begin{tikzcd} \Delta \arrow[r, "M;N"] \arrow[d, dotted, "a"'] \arrow[dr, phantom, "\alpha *_h \beta"] & \Xi \arrow[d, dotted, "c"] \\ \Delta' \arrow[r, "M';N'"'] & \Xi' \end{tikzcd}$$

and similarly for vertical composition. Each composition law $*_h$ and $*_v$ induces a category

$(\mathcal{D}_h, *_h, 1_-)$ and $(\mathcal{D}_v, *_v, 1^-)$ on cells, with identities:

$$
\begin{tikzcd}
\Delta \arrow[r, "\mathrm{id}_\Delta"] \arrow[d, dotted, "a"'] \arrow[dr, phantom, "1_a"] & \Delta \arrow[d, dotted, "a"] \\
\Delta' \arrow[r, "\mathrm{id}_{\Delta'}"'] & \Delta'
\end{tikzcd}
\qquad
\begin{tikzcd}
\Delta \arrow[r, "M"] \arrow[d, dotted, "\mathrm{id}^\Delta"'] \arrow[dr, phantom, "1^M"] & \Gamma \arrow[d, dotted, "\mathrm{id}^\Gamma"] \\
\Delta \arrow[r, "M"'] & \Gamma
\end{tikzcd}
$$

Four coherence properties must be verified by $\mathcal{D}$:

1. in all diagrams:

$$
\begin{tikzcd}
\Delta \arrow[r, "M"] \arrow[d, dotted, "a"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[r, "N"] \arrow[d, dotted, "b"] \arrow[dr, phantom, "\beta"] & \Xi \arrow[d, dotted, "c"] \\
\Delta' \arrow[r, "M'"] \arrow[d, dotted, "a'"'] \arrow[dr, phantom, "\gamma"] & \Gamma' \arrow[r, "N'"] \arrow[d, dotted, "b'"] \arrow[dr, phantom, "\delta"] & \Xi' \arrow[d, dotted, "c'"] \\
\Delta'' \arrow[r, "M''"'] & \Gamma'' \arrow[r, "N''"'] & \Xi''
\end{tikzcd}
\tag{7}
$$

   the result of composing first horizontally and then vertically is the same as the result of composing first vertically and then horizontally. Expressed as commutative diagrams:

$$
\begin{tikzcd}
\Delta \arrow[r, "M"] \arrow[d, dotted, "a"'] \arrow[drr, phantom, "\alpha *_h \beta"] & \Gamma \arrow[r, "N"] & \Xi \arrow[d, dotted, "c"] \\
\Delta' \arrow[r, "M'"] \arrow[d, dotted, "a'"'] \arrow[drr, phantom, "\gamma *_h \delta"] & \Gamma' \arrow[r, "N'"] & \Xi' \arrow[d, dotted, "c'"] \\
\Delta'' \arrow[r, "M''"'] & \Gamma'' \arrow[r, "N''"'] & \Xi''
\end{tikzcd}
=
\begin{tikzcd}
\Delta \arrow[r, "M"] \arrow[d, dotted, "a"'] & \Gamma \arrow[r, "N"] \arrow[d, dotted, "b"] & \Xi \arrow[d, dotted, "c"] \\
\Delta' \arrow[r, phantom, "\alpha *_v \gamma"] \arrow[d, dotted, "a'"'] & \Gamma' \arrow[r, phantom, "\beta *_v \delta"] \arrow[d, dotted, "b'"] & \Xi' \arrow[d, dotted, "c'"] \\
\Delta'' \arrow[r, "M''"'] & \Gamma'' \arrow[r, "N''"'] & \Xi''
\end{tikzcd}
$$

2. horizontal identities are preserved by composition:

$$
\begin{tikzcd}
\Delta \arrow[r, "M"] \arrow[d, dotted, "\mathrm{id}^\Delta"'] \arrow[dr, phantom, "1^M"] & \Gamma \arrow[r, "N"] \arrow[d, dotted, "\mathrm{id}^\Gamma"] \arrow[dr, phantom, "1^N"] & \Xi \arrow[d, dotted, "\mathrm{id}^\Xi"] \\
\Delta \arrow[r, "M"'] & \Gamma \arrow[r, "N"'] & \Xi
\end{tikzcd}
=
\begin{tikzcd}
\Delta \arrow[r, "M;N"] \arrow[d, dotted, "\mathrm{id}^\Delta"'] \arrow[dr, phantom, "1^{M;N}"] & \Xi \arrow[d, dotted, "\mathrm{id}^\Xi"] \\
\Delta \arrow[r, "M;N"'] & \Xi
\end{tikzcd}
$$

3. vertical identities are preserved by horizontal composition,
4. vertical horizontal identities and horizontal vertical identities coincide:

$$
\begin{tikzcd}
\Delta \arrow[r, "\mathrm{id}_\Delta"] \arrow[d, dotted, "\mathrm{id}^\Delta"'] \arrow[dr, phantom, "1_{\mathrm{id}^\Delta}"] & \Delta \arrow[d, dotted, "\mathrm{id}^\Delta"] \\
\Delta \arrow[r, "\mathrm{id}_\Delta"'] & \Delta
\end{tikzcd}
=
\begin{tikzcd}
\Delta \arrow[r, "\mathrm{id}_\Delta"] \arrow[d, dotted, "\mathrm{id}^\Delta"'] \arrow[dr, phantom, "1^{\mathrm{id}_\Delta}"] & \Delta \arrow[d, dotted, "\mathrm{id}^\Delta"] \\
\Delta \arrow[r, "\mathrm{id}_\Delta"'] & \Delta
\end{tikzcd}
$$

Double categories should be seen as extensions of 2-categories — which are double categories with identities as vertical arrows. 2-categories describe *computations* (no side-effects) ; double categories extend their scope to *interactions* (many side-effects).

A distributed implementation of the λ-calculus. Implementing a distributed λ-calculus (that is: make many computers work in different parts of the same λ-term, and interact) looks like a famous old magic trick. Pick a pretty girl in the audience (= the λ-term) put her in a wooden box (= the dynamics of β-elimination) and divide the box in two different parts (= the distributed implementation) with a sharp saw. Then, demonstrate to the audience that the girl behaves in two sections just as when she entered the receptacle a few minutes ago. For instance, tickle the toes on one side of the box, and make the head puff on the other side. Obviously, everybody in the audience admits that the two sections of the body live separated exactly as they did united. The only difference is that now the two sections *interact* from one box to another instead of *living together* in the same united body.

Double categories formalize this wizardry. Take a λ-term Q and divide it in two parts: a subterm P (= module) on one part, and the context $C[-]$ on the other, so that $Q = C[P]$. This manipulation may proceed in the syntactic category $\mathfrak{L}$. Typically, assume that $C[-]$ binds the variables $x_0, ..., x_{n-1}$ in P and that $\Delta, x_0 : A_0, ..., x_{n-1} : A_{n-1} \vdash P : B$. Then

$$\Delta \xrightarrow{C[P]} \Xi$$

factors as:

$$\Delta \xrightarrow{\lambda x_0 ... \lambda x_{n-1}.P} \Gamma \xrightarrow{C[yx_0 ... x_{n-1}]} \Xi$$

where Γ and Ξ are the contexts $(y : A_0 \to \cdots \to A_{n-1} \to B)$, and $z : C$.

Once the λ-term Q is factored (= divided), the double category lets the two sections $\lambda x_0 ... \lambda x_{n-1}.P$ and $C[x_0 \dots x_{n-1}]$ compute (= live) separately, and interact on their interface:

$$\begin{tikzcd}
\Delta \arrow[r, "\lambda x_0 ... \lambda x_{n-1}.P"] \arrow[d, dotted, "a"'] \arrow[dr, phantom, "\alpha_0"] & \Gamma \arrow[d, dotted, "b"] \\
\Delta' \arrow[r, "P'" description] \arrow[d, dotted, "a'"'] \arrow[dr, phantom, "\alpha_1"] & \Gamma' \arrow[d, dotted, "b'"] \\
\Delta'' \arrow[r, "P''" description] \arrow[d, dotted, "a''"'] & \Gamma'' \arrow[d, dotted, "b''"] \\
\Delta''' & \Gamma'''
\end{tikzcd}
\qquad
\begin{tikzcd}
\Gamma \arrow[r, "C[yx_1 ... x_n]"] \arrow[d, dotted, "b"'] \arrow[dr, phantom, "\beta_0"] & \Xi \arrow[d, dotted, "c"] \\
\Gamma' \arrow[r, "\mathbf{C}'" description] \arrow[d, dotted, "b'"'] \arrow[dr, phantom, "\beta_1"] & \Xi' \arrow[d, dotted, "c'"] \\
\Gamma'' \arrow[r, "\mathbf{C}''" description] \arrow[d, dotted, "b''"'] & \Xi'' \arrow[d, dotted, "c''"] \\
\Gamma''' & \Xi'''
\end{tikzcd}$$

The coherence laws of double categories implies that when composing the two separate computations $\alpha_0 *_v \cdots *_v \alpha_k$ on $\lambda x_0 ... \lambda x_{n-1}.P$ and $\beta_0 *_v \cdots *_v \beta_k$ on $C[yx_0 ... x_{n-1}]$ one obtains a diagram of the form:

$$\begin{tikzcd}
\Delta \arrow[r, "\lambda x_0 ... \lambda x_{n-1}.P"] \arrow[d, dotted, "a"'] \arrow[dr, phantom, "\alpha_0"] & \Gamma \arrow[r, "C[y x_0 ... x_{n-1}]"] \arrow[d, dotted, "b"] \arrow[dr, phantom, "\beta_0"] & \Xi \arrow[d, dotted, "c"] \\
\Delta' \arrow[r, "P'"] \arrow[d, dotted, "a'"'] \arrow[dr, phantom, "\alpha_1"] & \Gamma' \arrow[r, "\mathbf{C}'"] \arrow[d, dotted, "b'"] \arrow[dr, phantom, "\beta_1"] & \Xi' \arrow[d, dotted, "c'"] \\
\Delta'' \arrow[r, "P''"] \arrow[d, dotted, "a''"'] & \Gamma'' \arrow[r, "\mathbf{C}''"] \arrow[d, dotted, "b''"] & \Xi'' \arrow[d, dotted, "c''"] \\
\Delta''' & \Gamma''' & \Xi'''
\end{tikzcd}$$

hence a computation

$$\gamma_0 *_v \cdots *_v \gamma_k \quad = \quad (\alpha_0 *_h \beta_0) *_v \cdots *_v (\alpha_k *_h \beta_k)$$

of the composite $Q_0 = C[(\lambda x_0 ... \lambda x_{n-1}.P) x_0 ... x_{n-1}]$:

$$\begin{tikzcd}
\Delta \arrow[r, "Q_0"] \arrow[d, dotted, "a"'] \arrow[dr, phantom, "\gamma_0"] & \Xi \arrow[d, dotted, "c"] \\
\Delta' \arrow[r, "Q_1"] \arrow[d, dotted, "a'"'] \arrow[dr, phantom, "\gamma_1"] & \Xi' \arrow[d, dotted, "c'"] \\
\Delta'' \arrow[r, "Q_2"] \arrow[d, dotted, "a''"'] & \Xi'' \arrow[d, dotted, "c''"] \\
\Delta''' & \Xi'''
\end{tikzcd}$$

Conversely, modularity occurs when every cell γ from Q_0 is the composite of two cells α from $\lambda x_0 ... \lambda x_{n-1}.P$ and β from $C[y x_0 \ldots x_{n-1}]$. Modularity formalizes exactly our metaphor of *a woman divided with a saw.* Not all double categories enjoy the modularity property (this is not part of their definition) but all the double categories $\mathcal{D}$, $\mathcal{D}_{\mathbf{1},\perp}$ and $\mathcal{P}$ we construct in this article will be modular, see section 3.4 for details.

Linear logic. Example (6) illustrates that interpreting computations in a *modular* double category requires a flexible notion of λ-terms, in fact more flexible than λ-terms themselves. In particular, the procedure of substitution (information flow) should not be unidirectional any more, from left to right, but also very often from right to left, as witnessed by the type $A^\perp$ in example (6).

Quite fortunately, the work of Girard on linear logic and *proof-nets* spares us useless discussions on alternative ad'hoc syntaxes. Our work is yet another justification of the theoretical shift initiated in (Girard 1987) since after all, the traditional syntax of proofs (λ-terms) is too stiff to describe normalization as an interface communication — what we do here in a double category.

Categorical proof theory. We trace a brief history of categorical proof theory. The subject starts with the work of Lambek who constructs in (Lambek 1969) a (multi)-category in which the objects are logical formulas and the morphisms are equivalence classes of proofs. In this interpretation, the logical connectives become functor, the cut is modelled by composition, etc... The logic underlying (Lambek 1969) may be seen now as noncommutative intuitionistic multiplicative linear logic.

An important application of Lambek's idea came about in the early 70's in an article by Kelly and Mac Lane on the coherence problem for symmetric monoidal closed categories, see (Kelly MacLane 1971). A coherence theorem is a result which tells when diagrams commute in a free category of a given kind. Basically what is desired is a criterion for determining whether a diagram consisting of structural morphisms from (in this case) the theory of symmetric monoidal closed categories commutes. Previous results for monoidal and symmetric monoidal categories, due to Mac Lane had been combinatorial in nature, and failed in the closed case, see (MacLane 1971). The difficulty was solved by viewing the theory of symmetric monoidal closed categories as a deductive system — this is where Lambek's idea show up. Kelly and Mac Lane prove a cut-elimination theorem which says that every morphism which is a string of primitive morphisms is equivalent to a morphism in simpler form. Hence one can apply inductive techniques and establish coherence (in the unit-free case.) To each morphism f is assigned an involution on the set of literals,what (Blute 1993) calls the Kelly-Mac Lane graph of f. The authors show that two morphisms are equal iff they have the same Kelly-Mac Lane graph. Today, we understand that Kelly-Mac Lane graphs are really the axiom links of proof-nets in (unit-free) intuitionistic multiplicative linear logic. In particular, (Kelly MacLane 1971) shows how to compose Kelly-Mac Lane graphs, and proves that composition is always compatible, in the sense that no loops are made when forming the composite, which we see now as a consequence of the correctness criterion for proof-nets. See also (Kelly Laplaza 1980) for a similar work on compact closed categories.

The connection with linear logic came only later. In 1980, Barr wrote (Barr 1991) on $*$-autonomous categories. He was specifically interested in symmetric monoidal closed categories in which every object is isomorphic to its second dual. When a few years later he heard Girard talk about linear logic, Barr immediately realized that categorically, these were exactly $*$-autonomous categories plus extra stuff. This was the starting point for the interest for $*$-autonomous categories and especially the construction in the appendix of Barr's book, which we call now the Chu construction. The categorical axioms for linear logic appear for the first time in (Seely 1987) then (Bierman 1995) who refines them to consider cut-elimination.

In (Blute 1993), Blute sets out to prove coherence for $*$-autonomous categories, using Lambek's ideas. Lambek's result are rather unwieldy in that he is working in sequent calculus, which requires to give equations between sequent derivations and take into account commutative conversions. Blute recasts Lambek's work in terms of proof nets, instead of sequent derivations. So, a morphism in the free category is a cut-free proof net, rather than an equivalence class of sequent derivations. This reevaluates the previous work of (Kelly MacLane 1971) in proof-net terms and explicates the correspondence between Kelly-Mac Lane graphs and axiom links, the relation between compatibility and the Danos-Regnier criterion.

In this article, we construct double categories of typed and untyped nets, where we describe the syntactical interactions occurring between MLL proof-nets during cut-elimination, in a *modular* way. The category of Kelly-Mac Lane graphs (without MIX) discussed in (Blute 1993) reappears in our work as a subcategory of the category $\mathcal{P}$ of proof-nets, see sections 4.2 and 4.3.

Structure of the paper. Section 2 introduces the double category $\mathcal{D}$ of (untyped) well-formed nets: its horizontal category, its vertical category, and finally its cells. Section 3 establishes three properties of $\mathcal{D}$: that it is vertical (theorem 2), that it is reversible (theorem 3) and that it is modular (theorem 4). The appendix explains how to derive a "dynamic" semantics from the construction, in that case the geometry of interaction (Girard 1987b) for MLL proof-nets, and discusses the absence of cut and axiom links in the model.

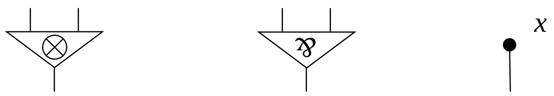
2. A double category for multiplicative nets

The section introduces a double category $\mathcal{D}$ modelling multiplicative (MLL) proof-nets. For simplicity, our presentation follows (Lafont 1995): we forget the units, the axiom and cut links, and focus on the kernel of linear logic consisting of the two connectives $\otimes$ (tensor) and $⅋$ (par). In a later section 4.4, we compare this presentation with the usual notion of proof-net (Girard 1987) containing axiom and cut links.

2.1. *Well-formed nets*

The first step toward $\mathcal{D}$ is to define a category $\mathcal{D}^h$ of syntactical *well-formed nets* and *well-formed modules.*

Nets. A net is a finite graph whose vertices are *tensor* ($\otimes$) links, *par* ($⅋$) links, and labelled *free ports*:



Each link has three *ports*: a *principal* one below and two *auxiliary* ones above. Each port of the net is connected to another port with a *wire* (= edge).

Well-formed nets. A *well-formed* net is defined inductively with the following rules:

1. a single wire connecting two free ports,
2. two well-formed nets connected by a single wire,
3. two well-formed nets connected by a tensor link through its auxiliary ports,
4. a well-formed net connected to itself by a par link through its auxiliary ports.

Proof-nets. The double category $\mathcal{D}$ will be a category of *untyped* nets. The reader

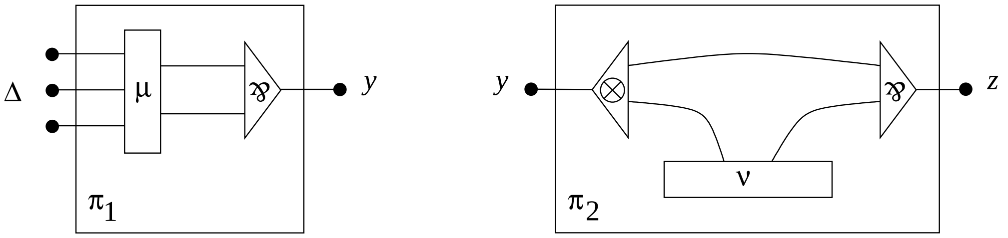


Fig. 1. The translations π_1 and π_2 of $(\lambda x.M)$ and $(\lambda w.yP)$.

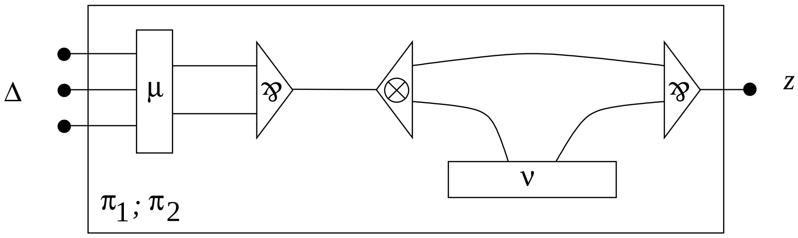


Fig. 2. The translation π of $(\lambda w.(\lambda x.M)P)$.

interested in their typed version, that is *proof-nets*, will find the definition in appendix, section 4.2, where a double category $\mathcal{P}$ of proof-nets is constructed.

2.2. *An horizontal category $\mathcal{D}^h$ of (well-formed) modules*

In our way toward $\mathcal{D}$, we construct its horizontal category $\mathcal{D}^h$. What category is $\mathcal{D}^h$? A first guess is that $\mathcal{D}^h$ has [lists of free ports] as objects and [well-formed nets] as morphisms. This is not sufficient though: as we will justify now, modularity implies that $\mathcal{D}^h$ admits also *non well-formed* nets as morphisms.

Let us illustrate this point with a variant of example (6). We consider two λ-terms M and P and their respective translation as well-formed nets μ and ν. The λ-terms $\lambda x.M$ and $\lambda w.yP$ translate respectively as the two nets π_1 and π_2 of figure 1. The λ-term $(\lambda w.(\lambda x.M)P)$ translates as their composite $\pi = \pi_1;\pi_2$ in figure 2. Following up the spirit of example (6), we would like to factor the cut-elimination cell $\alpha : \pi \Rightarrow_v \pi'$ as two cells $\alpha_1 : \pi_1 \Rightarrow_v \pi_1'$ and $\alpha_2 : \pi_2 \Rightarrow_v \pi_2'$ composing as $\alpha = \alpha_1 *_h \alpha_2$ in $\mathcal{D}$. The two nets π_1' and π_2' appear in figure 3. As required, they compose as $\pi_1';\pi_2' = \pi'$. Our main observation is that, contrary to π_1 and π_2, the net π_2' is *not* well-formed — its par link[§] is not connected twice to the component π. This illustrates this fact: that modularity of $\mathcal{D}$ implies the presence of non well-formed nets like π_2' as morphisms of $\mathcal{D}^h$.

[§] To explain the phenomenon in the λ-calculus, let us say that a λ-term $\lambda w.yP$ where the link λw binds a variable w in P, may transform into $\lambda w.z', x := P$ after a β-reduction, in the fashion of example (6). Of course, in $\lambda w.z', x := P$, the variable w appears outside the scope z' of λw.

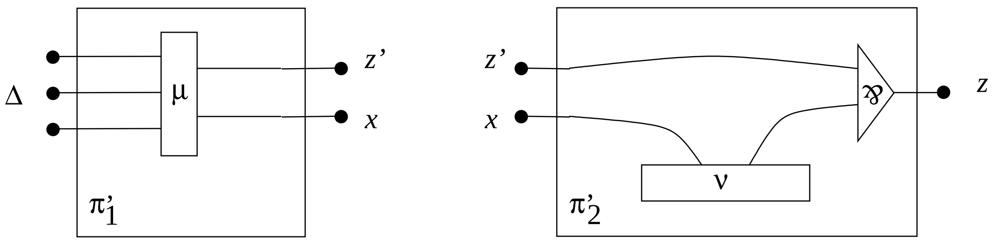
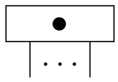


Fig. 3. The nets π_1 and π_2 after interaction.

One solution would be to consider that π_2' is well-formed in a logic enriched with the MIX rule, see (Fleury Rétoré 1994). In this article, we prefer the more satisfactory solution offered by the concept of module developped hereafter.

PARSING-NET. A *parsing box* is a link with a variable number of non-distinguished ports:



A *parsing-net* is a net possibly containing parsing boxes.

WELL-FORMED PARSING-NETS. Inspired by the long trip correctness criterion of (Girard 1987), (Danos Regnier 1989) characterize well-formed nets geometrically, as follows: a net π is well-formed if and only if every switching defines a tree (= connected acyclic graph). Here, a switching is the result of replacing all ⅋-links in π with one of the following patterns:

The definition of well-formedness is extended to parsing-nets in the obvious way: a parsing-net π is *well-formed* when every switching on π defines a connected acyclic graph.

ORTHOGONAL PARTITIONS. We recall that a partition e of the set $[n] = \{0, ..., n-1\}$ is a subset of $\wp[n]$ such that every element of $[n]$ is element of exactly one element of e. Alternatively, a partition of $[n]$ can be seen as a sequence of parsing boxes with free ports labelled by $0, ..., n-1$. We say that two partitions e and f of $[n]$ are orthogonal (à la Danos-Regnier) when the connection of their corresponding parsing-net is a well-formed parsing-net — which means here (there are no ⅋-link) that it is acyclic and connected. We write $e \perp f$.

$[m,n]$-NET. Let m and n be natural numbers. A $[m,n]$-net π is a net whose free ports are labelled by elements of $\{x_0, ..., x_{m-1}\} \cup \{y_0, ..., y_{n-1}\}$ in a one-to-one fashion.

PARTITION AND NET. Assume that π is a $[m,n]$-net, that e is a partition of $[m]$ and

f a partition of $[n]$. The parsing-net $e \bowtie \pi \bowtie f$ is the result of connecting the free ports $x_0, ..., x_{m-1}$ and $y_0, ..., y_{n-1}$ to (the parsing-nets corresponding to) e and f. The parsing-net $e \bowtie \pi$ is the result of connecting only e and π, and the parsing-net $\pi \bowtie f$ of connecting only f and π.

Assume that e is a partition of $[m]$. Every $[m,n]$-net π and switching s on π "transports" a partition e of $[m]$ to the partition f of $[n]$ which relates two indices $i, j \in [n]$ whenever the free ports y_i and y_j are connected in the parsing-net $e \bowtie \pi$ switched as s. Symmetrically, every $[m,n]$-net and switching on it transports a partition f of $[n]$ to a partition e of $[m]$. When the parsing-net $e \bowtie \pi \bowtie f$ is well-formed, every switching on π transports e to a partition orthogonal to f, and transports f to a partition orthogonal to e.

At this stage, we have accumulated enough concept and terminology to introduce the category $\mathcal{M}$ of *modules*, and derive the category $\mathcal{D}^h$ as a particular full subcategory of $\mathcal{M}$.

THE CATEGORY OF MODULES $\mathcal{M}$.

1. an object is a triple $\Delta = (m, E_\Delta, F_\Delta)$ where $m > 0$ is a natural number, E_Δ and F_Δ are two nonempty sets of partitions of $[m] = \{0, ..., m-1\}$ such that every $e \in E_\Delta$ and $f \in F_\Delta$ are orthogonal: we write $E_\Delta \perp F_\Delta$ in the situation above.
2. a morphism
$$\Delta = (m, E_\Delta, F_\Delta) \xrightarrow{\quad \pi \quad} \Gamma = (n, E_\Gamma, F_\Gamma)$$
is a $[m,n]$-net π which verifies for every $e \in E_\Delta$ and $f \in F_\Gamma$:
   (a) that $e \bowtie \pi \bowtie f$ is well-formed,
   (b) that every switching of π transports e to a partition in E_Γ,
   (c) that every switching of π transports f to a partition in F_Δ.
3. the identity id_Δ is the $[m,m]$-net consisting of m wires connecting each x_i and y_i, for $i \in [m]$,
4. the composite $\pi = \pi_1 ; \pi_2$ of $\pi_1 : \Delta \longrightarrow \Gamma$ and $\pi_2 : \Gamma \longrightarrow \Xi$ is the result of connecting the free port y_i in π_1 to the free port x_i in π_2 for every $i \in [n]$.

The definition of $\mathcal{M}$ deserves some explanations. The example opening this section 2 illustrates that $\mathcal{D}^h$ admits non well-formed nets as morphisms. But rather than *non well-formed nets*, these nets should be viewed as syntactic regions of larger *well-formed nets* — what (Girard 1987b) calls *modules*. The partitions in E_Δ and F_Γ are here to represent the different switchings of the environment of the module $\pi : \Delta \longrightarrow \Gamma$. Similar definitions appear in (Loader 1994; Abramsky Melliès 1999).

EMPTY OBJECTS. There is no empty object $\Xi = (0, \{\emptyset\}, \{\emptyset\})$ in $\mathcal{M}$. Such an object would contradict connectedness: two well-formed nets $\pi_1 : \Delta \longrightarrow \Xi$ and $\pi_2 : \Xi \longrightarrow \Gamma$ would compose as a non well-formed (non connected) net $\pi_1 ; \pi_2$. See the discussion in appendix, section 4.1.

MODULE. Arrows in $\mathcal{M}$ are called *modules*.

Remark: the category $\mathcal{M}$ is not "closed under cut-elimination". What we mean is that given a morphism $\pi : \Delta \longrightarrow \Gamma$ in $\mathcal{M}$ and a net π' obtained after a cut-elimination step on π, we do not have *necessarily* a morphism $\pi' : \Delta \longrightarrow \Gamma$ in $\mathcal{M}$. Why? Of course, the parsing-net $e \bowtie \pi' \bowtie f$ is well-formed for every $e \in E_\Delta$ and $f \in F_\Gamma$ because cut-elimination respects well-formedness, *but* every switching of π' does not necessarily transport every $e \in E_\Delta$ to a partition in E_Γ, or every $f \in F_\Gamma$ to a partition in F_Δ. In fact, the horizontal category $\mathcal{D}^h$ we are aiming at is not $\mathcal{M}$, but its full subcategory of *closed* objects.

Closed object. An object $\Delta = (m, E_\Delta, F_\Delta)$ is closed when:

1. every partition of $[m]$ orthogonal to every $e \in E_\Delta$ is in F_Δ,
2. every partition of $[m]$ orthogonal to every $f \in F_\Delta$ is in E_Δ.

We write $E_\Delta = (F_\Delta)^\perp$ and $F_\Delta = (E_\Delta)^\perp$ in the situation above.

Lemma 1. Suppose that $\Delta = (m, E_\Delta, F_\Delta)$ and $\Gamma = (n, E_\Gamma, F_\Gamma)$ are closed objects. Then, a $[m,n]$-net π is a morphism $\Delta \longrightarrow \Gamma$ in $\mathcal{M}$ if and only if the parsing-net $e \bowtie \pi \bowtie f$ is well-formed for every $e \in E_\Delta$ and $f \in F_\Gamma$.

Proof. ($\Rightarrow$) is direct. ($\Leftarrow$) Assume that $e \in E_\Delta$ and $f \in F_\Gamma$. Then $e \bowtie \pi \bowtie f$ is well-formed, thus every switching of π transports e to a partition e' orthogonal to f. Because Γ is closed and f runs over F_Γ, the partition e' is in E_Γ. This establishes that assertion (a) implies assertions (b) (and similarly (c)) in the definition of a module $\Delta \longrightarrow \Gamma$ in $\mathcal{M}$. We conclude. $\square$

So, when Δ and Γ are closed, and $\pi : \Delta \longrightarrow \Gamma$ is a module, every net π' obtained from π by cut-elimination is also a module $\pi' : \Delta \longrightarrow \Gamma$. This justifies the following definition.

The horizontal category $\mathcal{D}^h$.

1. its objects are the triples $\Delta = (m, E_\Delta, F_\Delta)$ where $m > 0$ is a natural number, E_Δ and F_Δ are nonempty sets of partitions of $[m]$ such that $E_\Delta = (F_\Delta)^\perp$ and $F_\Delta = (E_\Delta)^\perp$,
2. a morphism from $\Delta = (m, E_\Delta, F_\Delta)$ to $\Gamma = (n, E_\Gamma, F_\Gamma)$ is a $[m,n]$-module which verifies that $e \bowtie \pi \bowtie f$ is well-formed for every $e \in E_\Delta$ and $f \in F_\Gamma$.
3. composition and identities in $\mathcal{D}^h$ are defined as in $\mathcal{M}$.

Duality. Observe that $\mathcal{M}$ and $\mathcal{D}^h$ are self-dual with the endofunctor $(-)^{op}$ defined

1. on objects $\Delta = (m, E_\Delta, F_\Delta)$ as $\Delta^{op} = (m, F_\Delta, E_\Delta)$,
2. on morphisms $\pi : \Delta \longrightarrow \Gamma$ as $\pi^{op} : \Gamma^{op} \longrightarrow \Delta^{op}$ where the $[n,m]$-net π^{op} is π where each free port x_i (resp. y_j) is renamed as the free port y_i (resp. x_j), for every $i \in [m]$ and $j \in [n]$.

2.3. *The vertical category $\mathcal{D}^v$*

In our way towards $\mathcal{D}$, and after introducing its horizontal category $\mathcal{M}$, we define its the "vertical" category $\mathcal{D}^v$. The morphisms of this new category shall express interface interactions in the course of computation. We point out to the reader that the construction of $\mathcal{D}^v$ is motivated by the definition of cells in section 2.4. A glimpse at section 3.1 may be also useful at this stage.

$[m,n]$-FOREST. A $[m,n]$-forest is defined as a syntactical par-tensor forest with m roots and n leaves labelled by $x_0,...,x_{m-1}$ and $y_0,...,y_{n-1}$ respectively. Alternatively, a $[m,n]$-forest is a $[m,n]$-net containing no cut-elimination pattern.

A DUALITY ON TREES. Assume that a is a $[m,n]$-forest. The net $a^{\circledast}$ is defined as the forest a where every tensor (resp. par) link is replaced by a par (resp. tensor) link, and every free port x_i (resp. y_j) is replaced by y_i (resp. x_j).

THE VERTICAL CATEGORY $\mathcal{D}^v$.

1. its objects are the objects of $\mathcal{D}^h$,
2. a morphism from $\Delta = (m, E_\Delta, F_\Delta)$ to $\Gamma = (n, E_\Gamma, F_\Gamma)$ is a $[m,n]$-forest a which verifies for every $e_1 \in E_\Delta$, $e_2 \in F_\Delta$, $f_1 \in F_\Gamma$ and $f_2 \in E_\Gamma$, that $e_1 \bowtie a \bowtie f_1$ and $f_2 \bowtie a^{\circledast} \bowtie e_2$ are well-formed.
3. identity id^Δ on Δ is the trivial $[m,m]$-forest which wires x_i to y_i for every $i \in [m]$,
4. composition $(- \cdot -)$ is by connection.

Observe that the $[m,n]$-forest a is a vertical arrow from Δ to Γ precisely when (considered as the two $[m,n]$-nets a and $a^{\circledast}$) the $[m,n]$-forest a is a horizontal arrow from Δ to Γ and the $[m,n]$-forest $a^{\circledast}$ is a horizontal arrow from Γ to Δ. Categoricity of $\mathcal{D}^v$ follows.

INTERFACE. An arrow in $\mathcal{D}^v$ is called an *interface.*

Lemma 2. The lemma characterizes the interfaces from $\Delta = (m, E_\Delta, F_\Delta)$ to $\Gamma = (n, E_\Gamma, F_\Gamma)$. Assume that a is a $[m,n]$-forest. Every switching of a transports a partition e of $[m]$ to a partition e' of $[n]$. Let $E_\Delta \bowtie a$ denote the set of these partitions on $[n]$ when e runs over E_Δ. Define similarly $a^{\circledast} \bowtie F_a$. The following assertions are equivalent:

1. a is an interface $\Delta \longrightarrow_v \Gamma$,
2. a is a module $\Delta \longrightarrow_h \Gamma$ and $a^{\circledast}$ is a module $\Gamma \longrightarrow_h \Delta$,
3. $(E_\Delta \bowtie a) \perp F_\Gamma$ and $E_\Gamma \perp (a^{\circledast} \bowtie F_\Delta)$,
4. $F_\Gamma = (E_\Delta \bowtie a)^\perp$
5. $E_\Gamma = (a^{\circledast} \bowtie F_\Delta)^\perp$

Proof. Equivalence ($1 \iff 2$) is obvious, and has been already discussed. Before studying the remaining equivalences, we establish two orthogonality properties (8) and (9). Their proof is based on the fact that cut-elimination and η-expansion respect well-formedness of parsing-nets.

Assume that a is a $[m,n]$-forest and that $\Delta = (m, E_\Delta, F_\Delta)$ is an object of $\mathcal{D}^v$. We claim that

$$(E_\Delta \bowtie a) \perp (a^{\circledast} \bowtie F_\Delta) \tag{8}$$

Property (8) means that connecting the two parsing-nets $e \bowtie a$ and $a^{\circledast} \bowtie f$ on the n free ports of a and $a^{\circledast}$, for every $e \in E_\Delta$ and $f \in F_\Delta$, forms a well-formed net π. This net π, which could be noted $e \bowtie a \bowtie a^{\circledast} \bowtie f$ in a free notation, is the result of a sequence of η-expansions on the well-formed net $e \bowtie \mathbf{id}_\Delta \bowtie f$, see figure 4. Since

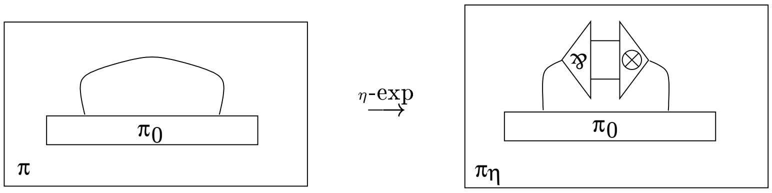


Fig. 4. η-expansion from π to π_η

applying each η-expansion respects well-formedness, by Danos-Regnier criterion, we may conclude orthogonality (8).

Assume now that a is an interface $\Delta \longrightarrow_v \Gamma$, assertion 1, or equivalently, that a is a module $\Delta \longrightarrow_h \Gamma$ and $a^{\circledast}$ is a module $\Gamma \longrightarrow_h \Delta$, assertion 2. We claim that

$$(E_\Delta \bowtie a)^\perp \perp (a^{\circledast} \bowtie F_\Delta)^\perp \tag{9}$$

Let f_1 and f_2 be any two partitions of $[n]$ such that, for every $e_1 \in E_\Delta$ and $e_2 \in F_\Delta$, the parsing-nets $e_1 \bowtie a \bowtie f_1$ and $f_2 \bowtie a^{\circledast} \bowtie e_2$ are well-formed. Assertion (9) states that f_1 and f_2 are orthogonal. In order to prove this, we consider that the parsing-net π, or $f_2 \bowtie a^{\circledast} \bowtie a \bowtie f_1$ in a free notation, obtained by connecting $f_2 \bowtie a^{\circledast}$ and $a \bowtie f_1$ together on the m free ports of a and $a^{\circledast}$. We prove that the net π is well-formed. By hypothesis on $e_1 \bowtie a \bowtie f_1$ for a given partition $e_1 \in E_\Delta$, no switching s of a induces a cycle in $a \bowtie f_1$. Symmetrically, no switching s' of $a^{\circledast}$ induces a cycle in $f_2 \bowtie a^{\circledast}$. Thus, well-formedness of π reduces to the fact that every switching s of a and s' of $a^{\circledast}$ transports f_1 and f_2 to *orthogonal* partitions f_1' and f_2' of $[m]$.

This last point follows from closure properties of Δ and Γ. By hypothesis, every switching of a transports f_1 to a partition f_1' orthogonal to every $e_1 \in E_\Delta$ — thus in F_Δ because Δ is closed. Similarly, every switching of $a^{\circledast}$ transports f_2 to a partition f_2' in E_Δ. Well-formedness of π follows from $E_\Delta \perp F_\Gamma$. To conclude the proof, the net π may be reduced by cut-elimination to the parsing-net $f_1 \bowtie \mathbf{id}_\Gamma \bowtie f_2$ — which is therefore well-formed. We obtain that $f_1 \perp f_2$, and assertion (9).

All in all, assertions (8) and (9) induce the following property[¶] when a is an interface $\Delta \longrightarrow_v \Gamma$:

$$(E_\Delta \bowtie a)^\perp = (a^{\circledast} \bowtie F_\Delta)^{\perp\perp} \qquad (a^{\circledast} \bowtie F_\Delta)^\perp = (E_\Delta \bowtie a)^{\perp\perp} \tag{10}$$

The end of the proof is elementary.
(2 $\iff$ 3) the direction (2 $\Rightarrow$ 3) is by definition, while direction (3 $\Rightarrow$ 2) follows from the forest structure of a and $a^{\circledast}$. Indeed, every two partitions $e \in E_\Delta$ and $f \in F_\Gamma$ define a net $e \bowtie a \bowtie f$. By the forest structure of a, whatever the switching s of a, the net $e \bowtie a$ accordingly switched does not contain any cycle. Thus, well-formedness of $e \bowtie a \bowtie f$ follows from orthogonality $(E_\Delta \bowtie a) \perp F_\Gamma$. Symmetrically, well-formedness

[¶] The property appears as a lemma in (Girard 1987b) section 4.5, stating that $\sigma(A^\perp)^\perp = \sigma(A)^{\perp\perp}$.

of $f \bowtie a^{\circledast} \bowtie e$ for every two partitions $f \in E_\Gamma$ and $e \in F_\Delta$, follows from orthogonality $E_\Gamma \perp (a^{\circledast} \bowtie F_\Delta)$.
($4 \Rightarrow 3$) Let a be a $[m,n]$-forest such that $F_\Gamma = (E_\Delta \bowtie a)^\perp$. Orthogonality $(E_\Delta \bowtie a) \perp F_\Gamma$ is obvious by hypothesis, while inclusion $(a^{\circledast} \bowtie F_\Delta) \subset F_\Gamma$, and therefore $(a^{\circledast} \bowtie F_\Delta) \perp E_\Gamma$, follows from property (8). We conclude.
($3 \Rightarrow 4$) By closure of Γ, assertion 3 implies that $(E_\Delta \bowtie a) \subset E_\Gamma$ and $(a^{\circledast} \bowtie F_\Delta) \subset F_\Gamma$. The first inclusion becomes $(E_\Delta \bowtie a)^\perp \supset F_\Gamma$ by dualization. The second inclusion becomes $(E_\Delta \bowtie a)^\perp \subset F_\Gamma$ by double dualization and assertion (10). We conclude that $F_\Gamma = (E_\Delta \bowtie a)^\perp$.
($3 \iff 5$) same proof as ($3 \iff 4$) modulo symmetry between a and $a^{\circledast}$. □

Remark: by lemma 2, every object $\Delta = (m, E_\Delta, F_\Delta)$ and $[m,n]$-forest a induce a unique object Γ such that $a : \Delta \longrightarrow_v \Gamma$. The object $\Gamma = (n, E_\Gamma, F_\Gamma)$ is defined by $F_\Gamma = (E_\Delta \bowtie a)^\perp$ and $E_\Gamma = (a^{\circledast} \bowtie F_\Delta)^\perp$.

2.4. *The double category $\mathfrak{D}$*

Pairing. A pairing $\mathfrak{P}$ on a set X is a symmetric relation which is one-to-one ($\forall (x,y,z) \in X^3, x\mathfrak{P}y$ and $x\mathfrak{P}z \Rightarrow y = z$) and fixpoint-free ($\forall x \in X, \neg(x\mathfrak{P}x)$).

Pairing on nets. Assume that π is a net and $\mathfrak{P}$ is a pairing on its (set of) links. Given two dual links L and L' forming a cut-elimination pattern in π, we say that $\mathfrak{P}$ *performs* their cut-elimination $\alpha : \pi \Rightarrow \pi'$ when $L\mathfrak{P}L'$. In that case, the *residual* $\mathfrak{P} \setminus \alpha$ of $\mathfrak{P}$ is the pairing of the links of π' resulting of $\mathfrak{P}$ after cut-elimination α.

By induction, we say that $\mathfrak{P}$ performs a sequence $\alpha_0, ..., \alpha_{m-1}$ of cut-elimination steps when $\mathfrak{P}$ performs $\alpha_0 : \pi_0 \Rightarrow \pi_1$ and $\mathfrak{P} \setminus \alpha$ performs the sequence $\alpha_1, ..., \alpha_{m-1}$. If $\mathfrak{P}$ performs the sequence $D = \alpha_0, ..., \alpha_{m-1}$, its residual noted $\mathfrak{P} \setminus D$ is defined as $\mathfrak{P} \setminus \alpha_0 \cdots \setminus \alpha_{m-1}$.

A pairing $\mathfrak{P}$ is declared *correct* when it performs a sequence $D : \pi \Rightarrow \pi_D$ which cut-eliminates all its paired links, ie. such that $\mathfrak{P} \setminus D$ is the empty pairing $\emptyset$. In that case, a simple "permutation" argument (adapted from the cube lemma in (Huet Lévy 1979)) shows that $\pi_D = \pi_E$ and $\mathfrak{P}' \setminus D = \mathfrak{P}' \setminus E$ for every sequence $E : \pi \Rightarrow \pi_E$ performed by $\mathfrak{P}$ such that $\mathfrak{P} \setminus E = \emptyset$, and every pairing $\mathfrak{P}' \supset \mathfrak{P}$ of π. We write $\pi \setminus \mathfrak{P}$ for π_D and say that $\pi \setminus \mathfrak{P}$ is the net obtained by cut-eliminating all the paired links in $\mathfrak{P}$. We also write $\mathfrak{P}' \setminus \mathfrak{P}$ for $\mathfrak{P}' \setminus D$.

Real vs Virtual. As a module, every vertical arrow $\Delta \longrightarrow_v \Gamma$ defines a pair of horizontal arrows $\Delta \longrightarrow_h \Gamma$ and $\Gamma \longrightarrow_h \Delta$, see lemma 2. To avoid unnecessary quiproquos between horizontal and vertical arrows, we call "real" the links in the former and "virtual" the links in the latter. So, every interface $a : \Delta \longrightarrow_v \Gamma$ is at the same time a "virtual" module $a : \Delta \longrightarrow_h \Gamma$ and a "virtual" module $a^{\circledast} : \Gamma \longrightarrow_h \Delta$.

Cells. A *cell* is a 5-uple $(\pi, \pi', a, b, \mathfrak{P})$ with two horizontal arrows $\pi : \Delta \longrightarrow_h \Gamma$ and $\pi' : \Delta' \longrightarrow_h \Gamma'$, two vertical vertical arrows $a : \Delta \longrightarrow_v \Delta'$ and $b : \Gamma \longrightarrow_v \Gamma'$, the four of them forming a "tile":

$$\begin{tikzcd}
\Delta \arrow[r, "\pi"] \arrow[d, dotted, "a"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "b"] \\
\Delta' \arrow[r, "\pi'"'] & \Gamma'
\end{tikzcd}$$

equipped with a pairing $\mathfrak{P}$ on the links of $a^{\circledast};\pi;b$ such that:

1. every virtual link is paired to another (virtual or real) link,
2. $\pi' = a^{\circledast};\pi;b \setminus \mathfrak{P}$ is the result of cut-eliminating all the paired links in $\mathfrak{P}$.

A cell α is characterised by its edges π, π', a, b and the real links paired by $\mathfrak{P}$. Observe that two cells with the same edges may differ for the same reason that there are two β-redexes $(\lambda x.x)((\lambda x.x)a) \longrightarrow (\lambda x.x)a$ in the λ-calculus.

Horizontal composition. We define the horizontal composite $\gamma = \alpha *_h \beta$ of two cells $\alpha = (\pi, \pi', a, b, \mathfrak{P}_\alpha)$ and $\beta = (\mu, \mu', b, c, \mathfrak{P}_\beta)$ as the cell $(\pi;\pi', \mu;\mu', a, c, \mathfrak{P}_\gamma)$ where $\mathfrak{P}_\gamma$ pairs two links L and L' in $a^{\circledast};\pi;\mu;c$ if and only if[‖]:

— $L\mathfrak{P}_\alpha L'$, or
— $L\mathfrak{P}_\beta L'$, or
— $L\mathfrak{P}_\alpha L_1$ and $L_2\mathfrak{P}_\beta L'$ for two links $L_1 \in b$ and $L_2 \in b^{\circledast}$ equal up to $(-)^{\circledast}$ duality.

Lemma 3. $*_h$ defines a category of cells.

Proof. Associativity of $*_h$ follows its definition as relational composition. The identities 1_a are given by $(\mathrm{id}_\Delta, \mathrm{id}_{\Delta'}, a, a, \mathfrak{P}_{1_a})$ where $\mathfrak{P}_{1_a}$ is the maximal correct pairing on $a^{\circledast};a$. We prove now that the pairing $\mathfrak{P}_\gamma$ defined hereabove is a correct pairing, so that $\gamma = \alpha *_h \beta$ defines a cell. We observe that the pairing $\mathfrak{P}_0 = \mathfrak{P}_\alpha + \mathfrak{P}_\beta$ is correct on

$$\pi_0 = a^{\circledast};\pi;b;b^{\circledast};\mu;c$$

and that the pairing $\mathfrak{P}_\gamma = \mathfrak{P}_k$ on $\pi_k = a^{\circledast};\pi;c$ is the result of a sequence η-reductions

$$(\pi_0, \mathfrak{P}_0) \rightsquigarrow (\pi_1, \mathfrak{P}_1) \rightsquigarrow \cdots \rightsquigarrow (\pi_k, \mathfrak{P}_k) \tag{11}$$

where by η-reduction $(\pi, \mathfrak{P}) \rightsquigarrow (\pi', \mathfrak{P}')$ we mean

— the elimination in π of an η-pattern L, L' where $\mathfrak{P}$ pairs L to $L_1 \neq L'$ and L' to $L_2 \neq L$, i.e. the converse of η-expansion in figure 4,
— on the resulting net π' the replacement of $\mathfrak{P}$ by its restriction $\mathfrak{P}'$ to π' where in addition the links L_1 and L_2 are paired.

We prove that $(\pi, \mathfrak{P}) \rightsquigarrow (\pi', \mathfrak{P}')$ respects the correctness of pairings, and conclude with sequence (11) that $\gamma = \alpha *_h \beta$ is really a cell. Indeed, assume that $\mathfrak{P}$ is a correct pairing. We show that $\mathfrak{P}'$ is correct by induction on the size of $\mathfrak{P}$, which is never zero. Let L and L' be the two links eliminated in the η-pattern and $\alpha : \pi \Rightarrow_v \mu$ be a cut-elimination step performed by $\mathfrak{P}$. If α does not involve L and L', then $\mathfrak{P}'$ performs the cut-elimination step $\alpha' : \pi' \Rightarrow_v \mu'$ corresponding to α after $(\pi, \mathfrak{P}) \rightsquigarrow (\pi', \mathfrak{P}')$, and there is an η-reduction

‖ Here, we identify the link L in $a^{\circledast};\pi;\mu;c$ with the corresponding link in $a^{\circledast};\pi;b$ or in $b^{\circledast};\mu;c$.

$(\mu, \mathfrak{P} \setminus \alpha) \rightsquigarrow (\mu', \mathfrak{P}' \setminus \alpha')$ corresponding to $(\pi, \mathfrak{P}) \rightsquigarrow (\pi', \mathfrak{P}')$ after α. Because $\mathfrak{P}' \setminus \alpha'$ is correct by induction hypothesis, $\mathfrak{P}'$ is correct. Now, if α involves L or L', it does not involve both for acyclicity reasons. It is easy to see then that $\pi' = \mu$ and that $\mathfrak{P}' = \mathfrak{P} \setminus \alpha$. From that, it follows that $\mathfrak{P}'$ is a correct pairing (either we choose α carefully hereabove so that $\mathfrak{P} \setminus \alpha$ is correct, for instance α starts a sequence D of cut-eliminations such that $\mathfrak{P} \setminus D = \emptyset$, or we take the time to prove that $\mathfrak{P} \setminus \alpha$ is correct for every cut-elimination step α performed by $\mathfrak{P}$.) $\square$

Vertical composition. We define the vertical composite $\gamma = \alpha *_v \beta$ of two cells $\alpha = (\pi, \pi', a, b, \mathfrak{P}_\alpha)$ and $\beta = (\pi', \pi'', a', b', \mathfrak{P}_\beta)$ as the cell $(\pi, \pi'', a \cdot a', b \cdot b', \mathfrak{P}_\gamma)$ where $\mathfrak{P}_\gamma \supset \mathfrak{P}_\alpha$ is the unique pairing such that $\mathfrak{P}_\gamma \setminus \mathfrak{P}_\alpha = \mathfrak{P}_\beta$.

Lemma 4. $*_v$ defines a category of cells.

Proof. By definition of $\mathfrak{P}_\gamma$, the 5-uple $\alpha *_v \beta$ is a cell. Moreover, the pairing $\mathfrak{P}_\gamma$ is (up to some link identification) $\mathfrak{P}_\alpha \cup \mathfrak{P}_\beta$. Associativity of $*_v$ follows. The identities 1^π are given by $(\pi, \pi, \mathrm{id}^\Delta, \mathrm{id}^\Gamma, \mathfrak{P}_{1^a})$ with the empty pairing $\mathfrak{P}_{1^a}$ on π. $\square$

The double category $\mathcal{D}$. We summarize our construction.

1. its horizontal category is $\mathcal{D}^h$ with composition $\pi; \mu$ and identities id_Δ,
2. its vertical category is $\mathcal{D}^v$ with composition $a \cdot b$ and identities id^Δ,
3. its cells α are the 5-uples $(\pi, \pi', a, b, \mathfrak{P})$ where $\mathfrak{P}$ is a pairing on $a^{\circledast}; \pi; b$ such that every virtual link is paired and π' is the result of cut-eliminating all the pairs in $\mathfrak{P}$,
4. cells are horizontally composed with $(- *_h -)$ and identities 1_a, and vertically composed with $(- *_v -)$ and identities 1^π.

Theorem 1 (Correctness). $\mathcal{D}$ is a double category.

Proof. We know that $\mathcal{D}^h$ and $\mathcal{D}^v$ form categories, and by lemmas 3 and 4 that $*_h$ and $*_v$ define categories on cells. There remains only to prove the four coherence conditions. Conditions 2. 3. and 4. are immediate. Condition 1. is almost immediate after the remark that a cell $(\pi, \pi', a, b, \mathfrak{P})$ is characterized by (π, π', a, b) and $\mathfrak{P}$ restricted to its real links. Thus, condition 1. reduces to the fact that every real link paired in $\mathfrak{P}_{(\alpha *_h \beta) *_v (\gamma *_h \delta)}$ is also paired in $\mathfrak{P}_{(\alpha *_v \gamma) *_h (\beta *_v \delta)}$ (see diagram (7), section 1 for notation). $\square$

3. The double category $\mathcal{D}$ is vertical, reversible and modular

After introducing elementary cells in section 3.1, we establish three important properties of the double category $\mathcal{D}$:

1. in section 3.2, the verticalization theorem states that every cell in $\mathcal{D}$ is the vertical composite of a finite sequence of "elementary" cells,
2. in section 3.3, the reversibility theorem connects horizontal and vertical arrows in $\mathcal{D}$, and opens the way to a "full completeness" result in section 4,
3. in section 3.4, the modularity theorem relates the double category $\mathcal{D}$ to the problematics of section 1.

3.1. *Elementary cells*

A cell $(\pi, \pi', a, b, \mathfrak{P})$ is declared elementary when $\mathfrak{P}$ pairs at most one couple (L, L') of tensor-par links in $a^{\circledast}; \pi; b$. Depending on whether L and L' are real or virtual links, the elementary cells part in four classes:

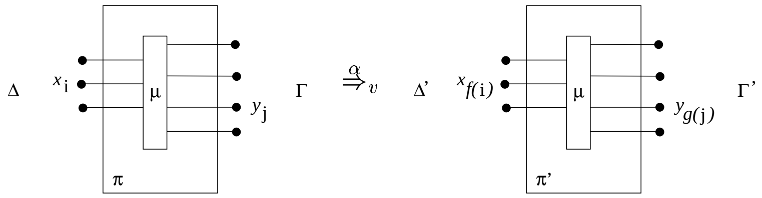
1. an isomorphic cell does not pair any link,
2. a "cut-elimination" cell pairs real with real,
3. an "emission" cell pairs real with virtual,
4. a "transmission" cell pairs virtual with virtual.

Letting $\Delta = (m, E_\Delta, F_\Delta)$ and $\Gamma = (n, E_\Gamma, F_\Gamma)$, we review the elementary double cells $\alpha = (\pi, \pi', a, b, \mathfrak{P})$ possibly transforming a module $\pi : \Delta \longrightarrow \Gamma$ into a module $\pi' : \Delta' \longrightarrow \Gamma'$:

$$\begin{tikzcd} \Delta \arrow[r, "\pi"] \arrow[d, dotted, "a"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "b"] \\ \Delta' \arrow[r, "\pi'"'] & \Gamma' \end{tikzcd}$$

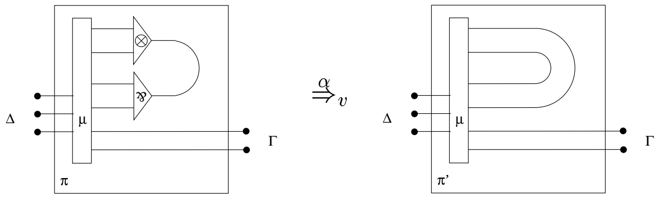
Renaming: every isomorphic cell α is characterized by the permutations f on $[m]$ and g on $[n]$ it operates on the labelling of π's free ports. Typically, in the figure below, $x_i \longmapsto x_{f(i)}$ and $y_j \longmapsto y_{g(j)}$ for every $i \in [m]$ and $j \in [n]$:

$$\begin{tikzcd} \Delta \arrow[r, "\pi"] \arrow[d, dotted, "\mathrm{id}^{\Delta}"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "\mathrm{id}^{\Gamma}"] \\ \Delta' \arrow[r, "\pi'"'] & \Gamma' \end{tikzcd}$$ for



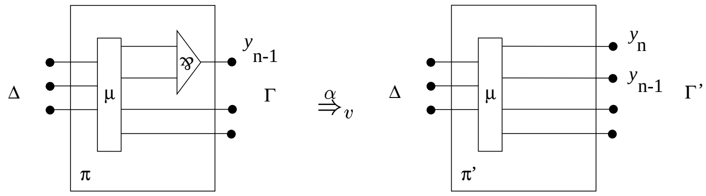
Cut-elimination: a cut-elimination cell α pairs two dual links in the module $\pi : \Delta \longrightarrow \Gamma$. Consequently, the edges a and b are the vertical identities on the domain Δ and codomain Γ of the module π (this is subject reduction).

$$\begin{tikzcd} \Delta \arrow[r, "\pi"] \arrow[d, dotted, "\mathrm{id}^{\Delta}"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "\mathrm{id}^{\Gamma}"] \\ \Delta \arrow[r, "\pi'"'] & \Gamma \end{tikzcd}$$ for



Emission: emission cells pair a real link in π with a virtual link in $a^{\circledast}$ or b. They are a consequence of modularity, in the following sense. A module π may compose horizontally with a module $\nu : \Gamma \longrightarrow_h \Xi$ in such a way that a cut-elimination pattern not in π or ν appears in the composite $\pi; \nu$, typically when a par link L in π points to a tensor link L' in ν. In that case, modularity requires that there exists a cell $\alpha : \pi \Rightarrow_v \pi'$ which interprets (= factors) the cut-elimination step occurring in $\pi;\nu$, see section 3.4. This cell is an emission cell, which "emits" the par link L:

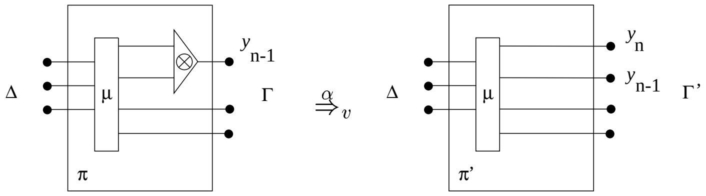
$$\begin{tikzcd} \Delta \arrow[r, "\pi"] \arrow[d, dotted, "\mathrm{id}^{\Delta}"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "\parr_n"] \\ \Delta \arrow[r, "\pi'"'] & \Gamma' \end{tikzcd} \quad \text{for}$$



Here, the vertical arrow $\parr_n$ is defined as the $[n, n+1]$-forest with a *tensor* link pointing its principal port to x_{n-1} and its auxiliary ports to y_{n-1} and y_n, and $n-1$ wires connecting each x_i to its corresponding y_i for $i \in [n-1]$. The pairing $\mathfrak{P}$ on $\pi;\parr_n$ pairs the real par link L in π with the virtual tensor link in $\parr_n$.

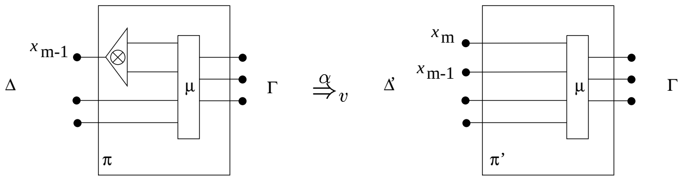
The dual emission cell α "emits" a tensor link L. It is obtained by pairing in $\pi;\otimes_n$ the virtual par link in $\otimes_n = (\parr_n^{\circledast})^{op}$ with the real tensor link L in π.

$$\begin{tikzcd} \Delta \arrow[r, "\pi"] \arrow[d, dotted, "\mathrm{id}^{\Delta}"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "\otimes_n"] \\ \Delta \arrow[r, "\pi'"'] & \Gamma' \end{tikzcd} \quad \text{for}$$

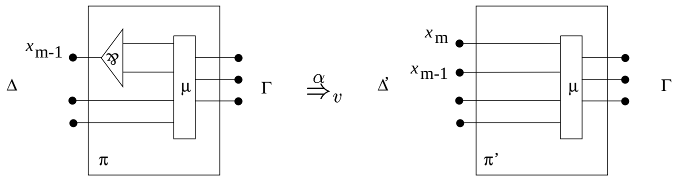


After the elementary cells emitting on their codomain, we review the elementary cells emitting on their domain, which for that reason interact (= horizontally compose) with the previous ones.

$$\begin{tikzcd} \Delta \arrow[r, "\pi"] \arrow[d, dotted, "\parr_m"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "\mathrm{id}^{\Gamma}"] \\ \Delta' \arrow[r, "\pi'"'] & \Gamma \end{tikzcd} \quad \text{for}$$



$$\begin{tikzcd} \Delta \arrow[r, "\pi"] \arrow[d, dotted, "\otimes_m"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "\mathrm{id}^{\Gamma}"] \\ \Delta' \arrow[r, "\pi'"'] & \Gamma \end{tikzcd} \quad \text{for}$$



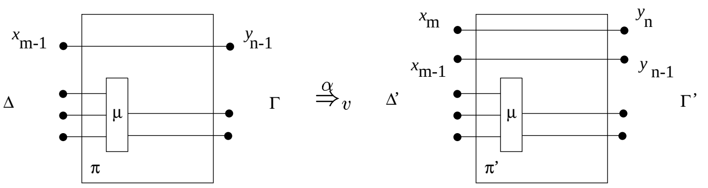
Now, the introductory example of section 2.2, figures 1, 2 and 3, illustrates the case of two emission cells $\alpha_1 : \pi_1 \Rightarrow_v \pi'_1$ and $\alpha_2 : \pi_2 \Rightarrow_v \pi'_2$ composing horizontally as a cut-elimination cell $\alpha_1 *_h \alpha_2 : \pi_1;\pi_2 \Rightarrow_v \pi'_1;\pi'_2$.

Transmission: Transmission cells pair two virtual links in $a^{\circledast}$ and b connected by a wire in π. Like emission cells, transmission cells are required by modularity. Suppose that three modules μ, π and ν compose in such a way that a tensor link L in $\mu : \Theta \longrightarrow \Delta$, a par link L' in $\nu : \Delta \longrightarrow \Xi$ and a wire in $\pi : \Gamma \longrightarrow \Delta$ connecting L with L' form a cut-elimination pattern in $\mu;\pi;\nu : \Theta \longrightarrow \Xi$.

$$\Theta \xrightarrow{\mu} \Delta \xrightarrow{\pi} \Gamma \xrightarrow{\nu} \Xi$$

Modularity implies that the cut-elimination cell on $\mu;\pi;\nu$ is reflected (=factored) as cells on μ, π and ν. Two emission cells $\mu \Rightarrow_v \mu'$ and $\nu \Rightarrow_v \nu'$ in μ and ν and a transmission cell α in π play that role. The transmission cell is of the form:

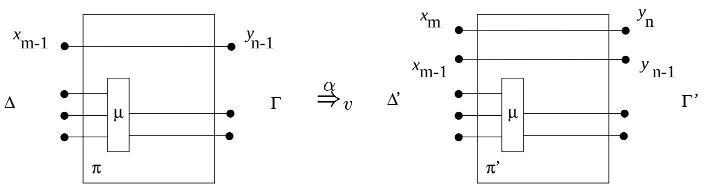
$$\begin{tikzcd} \Delta \arrow[r, "\pi"] \arrow[d, dotted, "\otimes_m"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "\otimes_n"] \\ \Delta' \arrow[r, "\pi'"'] & \Gamma' \end{tikzcd}$$ for



The cell is constructed by pairing in $\otimes_m^{\circledast};\pi;\otimes_n$ the virtual tensor link in $\otimes_m^{\circledast}$ with the virtual par link in $\otimes_n$. In the course of transmission, the wire in π divides in two wires which connect the resulting μ' and ν'. In a typed setting, this demultiplexing transforms the wire of type $A \otimes B$ into two wires of type A and B.

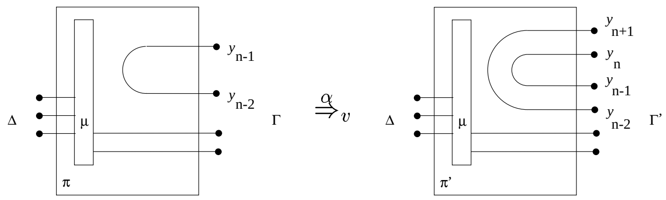
A symmetric cell is required in $\mathcal{D}$ to transmit the emission of $\otimes$ on Δ to the emission of $\parr$ on Γ:

$$\begin{tikzcd} \Delta \arrow[r, "\pi"] \arrow[d, dotted, "\parr_m"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "\parr_n"] \\ \Delta' \arrow[r, "\pi'"'] & \Gamma' \end{tikzcd}$$ for



Yet another possibility of transmission occurs when a wire in $\pi : \Delta \longrightarrow \Gamma$ connects two dual links L and L' of the same module $\nu : \Gamma \longrightarrow \Xi$. In that case, the transmission cell is as follows:

$$\begin{tikzcd} \Delta \arrow[r, "\pi"] \arrow[d, dotted, "\mathrm{id}^{\Delta}"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "(\parr\otimes)_n"] \\ \Delta \arrow[r, "\pi'"'] & \Gamma' \end{tikzcd}$$ for

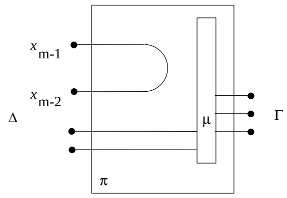


Here, $(\parr\otimes)_n$ is the $[n, n+2]$-forest with a tensor link L pointing its principal port to x_{n-2} and its auxiliary ports to y_{n-2} and y_{n-1}, and a par link L' pointing its principal port to x_{n-1} and its auxiliary ports to y_{n+1} and y_n, and n wires connecting each x_i to its corresponding y_i for $i \in [n-1]$. The cell α pairs the two virtual links L and L' in $\pi;(\parr\otimes)_n$.

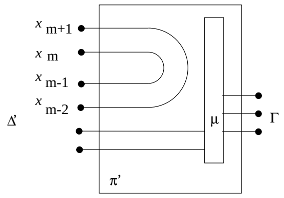
We conclude our gallery of elementary cells in $\mathcal{D}$ with the cell which transmits on its domain.

$$\begin{tikzcd} \Delta \arrow[r, "\pi"] \arrow[d, dotted, "(\parr\otimes)_m"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "\mathrm{id}^\Gamma"] \\ \Delta' \arrow[r, "\pi'"'] & \Gamma \end{tikzcd}$$

for



$\overset{\alpha}{\Rightarrow}_v$



3.2. *The verticalization theorem*

Verticalization expresses a *synchronization* property of computations in the double category $\mathcal{D}$. We say that a cell $\alpha : \pi \Rightarrow_v \pi'$ is *vertical* when it is a sequence of “elementary actions” on π, in other words when there exists a sequence $\alpha_1, ..., \alpha_k$ of *elementary* cells such that:

$$\alpha = \alpha_1 *_v \cdots *_v \alpha_k$$

The verticalization theorem states that every cell α in $\mathcal{D}$ is vertical. This means that whenever two vertical cells α and β:

$$\alpha = \alpha_1 *_v \cdots *_v \alpha_k \qquad \beta = \beta_1 *_v \cdots *_v \beta_l$$

compose *horizontally* on their interface b:

$$\alpha : a \Rightarrow_h b \text{ and } \beta : b \Rightarrow_h c$$

the independent sequences of “elementary actions” $(\alpha_1, ..., \alpha_k)$ and $(\beta_1, ..., \beta_l)$ *synchronize* to compute the horizontal composite $\gamma = \alpha *_h \beta$ as a *vertical* sequence $(\gamma_1, ..., \gamma_n)$ of elementary cells:

$$\gamma = \gamma_1 *_v \cdots *_v \gamma_n$$

Theorem 2 (verticalization). Every cell $\gamma : \pi \Rightarrow_v \pi'$ is the vertical composite $\gamma_0 *_v \cdots *_v \gamma_{n-1}$ of a sequence of elementary cells $\gamma_i : \pi_i \Rightarrow_v \pi_{i+1}$ with $\pi = \pi_0$ and $\pi_n = \pi'$.

Proof. Let $\gamma = (\pi, \pi', a, b, \mathfrak{P}_\gamma)$ be any cell in the double category $\mathcal{D}$. The proof goes by induction on the size of the pairing $\mathfrak{P}_\gamma$ on $\mu = a^\circledast; \pi; b$. We conclude when $\mathfrak{P}_\gamma$ is empty. Otherwise, $\mathfrak{P}_\gamma$ performs at least the cut-elimination of a pair (L, L') in μ. A *prefix* of the interface a is any interface a_0 such that a factors as $a = a_0 \cdot a'$ in $\mathcal{D}^v$. Let a_0 and b_0 be the minimal prefixes of a and b such that L and L' appear in the subnet $\mu_0 = a_0^\circledast; \pi; b_0$ of $\mu = a^\circledast; \pi; b$. The cell $\gamma_0 = (\pi, \pi_1, a_0, b_0, \mathfrak{P}_0)$ pairing L and L' is elementary: it is a cut-elimination cell when L and L' are real, a transmission cell when L and L' are virtual, and an emission cell otherwise. Moreover, the cell γ factors as $\gamma = \gamma_0 *_v \gamma'$ where the cell γ' is defined as $\gamma' = (\pi_1, \pi', a', b', \mathfrak{P}_{\gamma'})$ where $\mathfrak{P}_{\gamma'} = \mathfrak{P}_\gamma \setminus \alpha$.

$$\begin{tikzcd} \Delta \arrow[r, "\pi"] \arrow[d, dotted, "a_0"'] \arrow[dr, phantom, "\gamma_0"] & \Gamma \arrow[d, dotted, "b_0"] \\ \Delta_1 \arrow[r, "\pi_1"'] & \Gamma_1 \end{tikzcd} \qquad \begin{tikzcd} \Delta_1 \arrow[r, "\pi_1"] \arrow[d, dotted, "a'"'] \arrow[dr, phantom, "\gamma'"] & \Gamma_1 \arrow[d, dotted, "b'"] \\ \Delta' \arrow[r, "\pi'"'] & \Gamma' \end{tikzcd}$$

We conclude the proof by induction on γ'. $\square$

Remark: the verticalization theorem may be easily reformulated as a *vertical* version of the modularity theorem in section 3.4.

3.3. *The reversibility theorem*

The reversibility theorem plays a central rôle in the full completeness result of section 4.3. It ensures that in the following situation:

$$\begin{tikzcd} \Delta \arrow[d, dotted, "a"'] & \Gamma \arrow[d, dotted, "b"] \\ \Delta' \arrow[r, "\pi'"'] & \Gamma' \end{tikzcd}$$

there exists a module $\pi : \Delta \longrightarrow_h \Gamma$ and a cell α of the form:

$$\begin{tikzcd} \Delta \arrow[r, "\pi"] \arrow[d, dotted, "a"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "b"] \\ \Delta' \arrow[r, "\pi'"'] & \Gamma' \end{tikzcd}$$

Theorem 3 (reversibility). Let $a : \Delta \longrightarrow_v \Delta'$, $b : \Gamma \longrightarrow_v \Gamma'$ and $\pi' : \Delta' \longrightarrow_h \Gamma'$. Then, there exists a module $\pi : \Delta \longrightarrow_h \Gamma$ and a cell α such that $\alpha : \pi \Rightarrow_v \pi'$ and $\alpha : a \Rightarrow_h b$.

Proof. Obvious. We define $\pi : \Delta \longrightarrow_h \Gamma$ as the module $a; \pi'; b^{\circledast}$, and $\mathfrak{P}$ as the pairing on $a^{\circledast}; \pi; b = (a^{\circledast}; a); \pi'; (b^{\circledast}; b)$ which pairs all the (symmetric) links in $a^{\circledast}; a$ and $b^{\circledast}; b$. The pairing $\mathfrak{P}$ is correct and $a^{\circledast}; \pi; b$ reduces to π' after cut-eliminating all the paired links in $\mathfrak{P}$. We conclude that the quadruple $\alpha = (\pi, \pi', a, b, \mathfrak{P})$ is a cell. $\square$

3.4. *The modularity theorem*

Modularity motivates the construction of double categorical models of computation, as already explained in section 1. The property states that for every cell γ of the form:

$$\begin{tikzcd} \Delta \arrow[r, "\mu"] \arrow[d, dotted, "a"'] & \Gamma \arrow[r, "\nu"] \arrow[d, phantom, "\gamma"] & \Xi \arrow[d, dotted, "c"] \\ \Delta' \arrow[rr, "\pi'"'] & & \Xi' \end{tikzcd}$$

there are two cells α and β

$$\begin{tikzcd} \Delta \arrow[r, "\mu"] \arrow[d, dotted, "a"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[d, dotted, "b"] \\ \Delta' \arrow[r, "\mu'"'] & \Gamma' \end{tikzcd} \qquad \begin{tikzcd} \Gamma \arrow[r, "\nu"] \arrow[d, dotted, "b"'] \arrow[dr, phantom, "\beta"] & \Xi \arrow[d, dotted, "c"] \\ \Gamma' \arrow[r, "\nu'"'] & \Xi' \end{tikzcd}$$

such that

$$\begin{array}{ccc}
\begin{tikzcd}
\Delta \arrow[r, "\mu"] \arrow[d, dotted, "a"'] \arrow[dr, phantom, "\alpha"] & \Gamma \arrow[r, "\nu"] \arrow[d, dotted, "b"] \arrow[dr, phantom, "\beta"] & \Xi \arrow[d, dotted, "c"] \\
\Delta' \arrow[r, "\mu'"'] & \Gamma' \arrow[r, "\nu'"'] & \Xi'
\end{tikzcd}
& = &
\begin{tikzcd}
\Delta \arrow[r, "\mu"] \arrow[d, dotted, "a"'] \arrow[drr, phantom, "\gamma"] & \Gamma \arrow[r, "\nu"] & \Xi \arrow[d, dotted, "c"] \\
\Delta' \arrow[rr, "\pi'"'] & & \Xi'
\end{tikzcd}
\end{array}$$

By the verticalization theorem (theorem 2) we need only to verify the property for elementary cells γ.

Theorem 4 (modularity). For every cell $\gamma : \mu;\nu \Rightarrow_v \pi'$ there are two cells $\alpha : \mu \Rightarrow_v \mu'$ and $\beta : \nu \Rightarrow_v \nu'$ such that $\gamma = \alpha *_h \beta$.

Proof. Clearly, every isomorphic cell γ verifies the modularity property, and we limit our attention to a non-isomorphic elementary cell $\gamma = (\mu;\nu,\pi',a,c,\mathfrak{P})$. By definition, $\mathfrak{P}_\gamma$ pairs a tensor link L with a par link L' in $\pi_0 = a^{\circledast};\mu;\nu;c$. The only difficult case occurs when the wire w which connects the principal ports of L and L' in π_0 is divided in $k+1$ wires $w_0,...,w_k$ of $\mu_0 = a^{\circledast};\mu$ and $\nu_0 = \nu;c$. Among the $k+1$ wires in μ_0 and ν_0, the wires $w_1,...,w_{k-1}$ connect a free port to a free port, while the wire w_0 (resp. w_k) connects the principal port of L (resp. L') to a free port. We may apply k η-expansions on the wire w in $\pi_0 = \mu_0;\nu_0$ in such a way that every new tensor link points its principal port to the principal port of a par node[††]. We obtain this way a net π_1 which factors in two nets μ_1 and ν_1 similar to μ_0 and ν_0, except that

— the wire w_0 is replaced by a par link pointing its principal port to L,
— the wire w_k is replaced by a tensor link pointing its principal port to L',
— every wire $w_1,...,w_{k-1}$ (connecting two free ports) is replaced by a cut-elimination pattern (connecting four free ports).

Now, it is easy to construct a $[k,2k]$-forest b such that $\mu_1 = \mu_0;b$ and $\nu_1 = b^{\circledast};\nu_0$, and to construct the two pairings $\mathfrak{P}_1$ and $\mathfrak{P}_2$ characterizing the cells $\alpha = (\mu,\mu',a,b,\mathfrak{P}_1)$ and $\beta = (\nu,\nu',b,c,\mathfrak{P}_2)$ we are looking for. $\square$

The modularity property is called *glueing lemma* in (Melliès 1994) and appears for the first time in its appropriate double categorical formulation in (Gadducci Montanari 1996) under the more diagrammatic name of *tile decomposition.*

4. Appendix

4.1. *Monoidal structures*

We discuss briefly the possibility of equipping our categories with a monoidal structure.

TWO DUAL EMPTY OBJECTS **1** AND $\perp$. In section 2.2, we forbid the existence of an "empty" object like $\Xi = (0,\{\emptyset\},\{\emptyset\})$ in the double category $\mathcal{D}$ because, we argue, this would contradict our restriction to well-connected modules. Yet, one may equip

[††] The orientation of the tensor-par links would be forced in a typed setting.

$\mathcal{D}$ with two *dual* empty objects, that is: a left object $\mathbf{1} = (0, \emptyset, \{\emptyset\})$ and a right object $\perp = (0, \{\emptyset\}, \emptyset)$. The previous contradiction with connectedness disappears because the resulting category $\mathcal{D}^h_{\mathbf{1},\perp}$ admits no morphism from $\perp$ to $A \neq \perp$, or from $A \neq \mathbf{1}$ to $\mathbf{1}$. Maps from $\mathbf{1}$ to $\perp$ are well-formed nets with no conclusion. Like $\mathcal{D}$, the resulting double category $\mathcal{D}_{\mathbf{1},\perp}$ is vertical, reversible and modular.

MONOIDAL STRUCTURES. It is a nice exercise to prove that given two closed objects $\Gamma = (m, E_\Gamma, F_\Gamma)$ and $\Delta = (n, E_\Delta, F_\Delta)$, the object $\Gamma \otimes \Delta = (m + n, E_{\Gamma\otimes\Delta}, F_{\Gamma\otimes\Delta})$ defined as follows

$$E_{\Gamma\otimes\Delta} = \{e_1 + e_2 \mid e_1 \in E_\Gamma, e_2 \in E_\Delta\} \qquad F_{\Gamma\otimes\Delta} = E^\perp_{\Gamma\otimes\Delta}$$

is closed too (hint: there are enough partitions in $F_{A\otimes B}$ to "detect" a partition in $E^{\perp\perp}_{\Gamma\otimes\Delta}$ connecting a port in Γ and a port in Δ). This defines a tensor product in the category of modules $\mathcal{D}^h$, unfortunately with no unit 1 in the category. At this point, moving to the category $\mathcal{D}^h_{\mathbf{1},\perp}$ does not solve anything, because strangely enough, $\mathcal{D}^h_{\mathbf{1},\perp}$ cannot be turned into a monoidal category. Indeed, tensoring two well-formed modules $A \longrightarrow \perp$ and $1 \longrightarrow B$ would induce a "non-connected" map $A \otimes 1 \longrightarrow \perp \otimes B$. A simple solution to overcome these difficulties and obtain a $*$-autonomous category is to add the MIX rule in the logic, in the spirit of (Blute 1993).

4.2. *Typing objects and modules in* $\mathcal{D}$

Modules in the double category $\mathcal{D}$ are "untyped". Here, we type them with formulas of (unit-free) multiplicative linear logic (MLL) and deduce a double category $\mathcal{P}$ of *proof-nets*.

TYPED NETS. A *typed net* is a net where each wire is labelled with a MLL formula. If a wire is explicitly labelled with A for a given orientation, it is also labelled implicitly with $A^\perp$ in the opposite direction:

The connecting wires are typed as follows:

PROOF-NETS. A *proof-net* is defined as a typed well-formed net.

TYPED OBJECTS. A *typed object* is a quadruple $\Gamma = (n, \mathtt{type}_\Gamma, E_\Gamma, F_\Gamma)$ where (n, E_Γ, F_Γ) is an object of $\mathcal{D}$ and $\mathtt{type}_\Gamma$ is a function which associates a MLL formula to every $i \in [n]$.

THE HORIZONTAL CATEGORY $\mathcal{P}^h$ OF PROOF-MODULES. Its objects are the typed objects plus $\mathbf{1}$ and $\perp$. Its morphisms $\Delta \longrightarrow \Gamma$ are the typed nets which (untyped) are morphisms (in $\mathcal{D}^h_{\mathbf{1},\perp}$) from Δ to Γ, and whose free port labelled x_i (resp. y_j) has formula $\mathtt{type}_\Delta(i)$ (resp. $\mathtt{type}^\perp_\Gamma(j)$). Identities and composition are defined as in $\mathcal{D}^h_{\mathbf{1},\perp}$.

PROOF-MODULE. An arrow in $\mathcal{P}^h$ is called a proof-module.

DUALITY. The duality functor $(-)^{op}$ of $\mathcal{D}^h$ and $\mathcal{D}^h_{\mathbf{1},\perp}$ extends to $\mathcal{P}^h$ by associating to every typed object $\Gamma = (n, \mathtt{type}_\Gamma, E_\Gamma, F_\Gamma)$ the typed object $\Gamma^{op} = (n, \mathtt{type}^\perp_\Gamma, F_\Gamma, E_\Gamma)$.

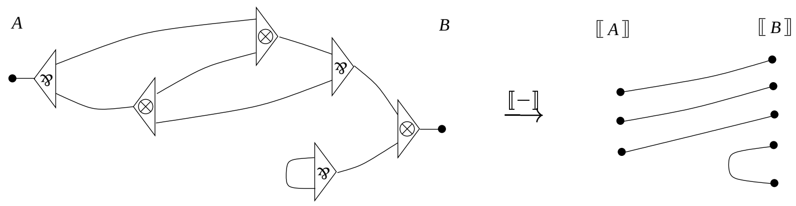
A DOUBLE CATEGORY $\mathcal{P}$ OF PROOF-MODULES. We leave the reader guess for himself the construction of the double category $\mathcal{P}$ of MLL proof-nets, obtained by "typing" the double category $\mathcal{D}$ (or more exactly $\mathcal{D}_{\mathbf{1},\perp}$). Obviously, this double category $\mathcal{P}$ has the horizontal category $\mathcal{P}^h$. Like $\mathcal{D}$ and $\mathcal{D}_{\mathbf{1},\perp}$, the double category $\mathcal{P}$ is vertical, reversible and modular, with variants of theorems 2, 3 and 4.

4.3. *Consumed forms*

One benefit of typing modules and moving from the double category $\mathcal{D}^h$ to the double category $\mathcal{P}$, is that the vertical category $\mathcal{P}^v$ of $\mathcal{P}$ is *strongly normalising.* This means that for every object $\Delta = (m, \mathtt{type}_\Delta, E_\Delta, F_\Delta)$ there is a bound on the number of emissions/transmissions on a vertical arrow $a : \Delta \longrightarrow_v \Delta'$. This bound may be expressed as $\sum_{i\in[m]} |\mathtt{type}_\Delta(i)|$, where $|A|$ denotes the number of connectives in the formula A. Moreover, $\mathcal{P}^v$ is locally confluent, see (Klop 1992) for a definition, thus every object Δ reduces vertically to a unique object $[\![A]\!]$ modulo $\mathcal{P}^v$-isomorphisms (= renamings).

CONSUMED OBJECTS AND MODULES. A *literal* is a formula with no tensor or par, thus of the form α or $\alpha^\perp$. We declare *consumed* any object like $\mathbf{1}$, $\perp$, or $(m, \mathtt{type}_\Delta, E_\Delta, F_\Delta)$ where $\mathtt{type}_\Delta(i)$ is a literal for every $i \in [m]$. A proof-module is *consumed* when it contains no links, only wires.

CONSUMED FORM. We have just associated to every formula A a consumed object $[\![A]\!]$. In a similar fashion, we associate to every proof-module $\pi : \Delta \longrightarrow_h \Gamma$ a *consumed form* $[\![\pi]\!] : [\![\Delta]\!] \longrightarrow_h [\![\Gamma]\!]$, defined as what remains of π after complete evacuation of the par and tensor links: therefore, its axiom links, what (Blute 1993) calls a Kelly-Mac Lane graph. We illustrate the definition with the transformation of $\pi : A \longrightarrow_h B$ into its consumed form $[\![\pi]\!] : [\![A]\!] \longrightarrow_h [\![B]\!]$:



A CATEGORY OF CONSUMED MODULES. We call $\mathcal{CONS}$ the subcategory of $\mathcal{P}^h$ whose objects and morphims are consumed.

Now, if we follow the analysis of (Girard 1987b), the consumed form of a module should retain its full semantical portrait. The tensor-par links are just the way semantics is "trapped" into syntax, i.e made to look like formulas. So, the following "full completeness" result should not be a surprise.

Theorem 5. $[\![-]\!]$ defines a full functor from $\mathcal{P}^h$ to $\mathcal{CONS}$. In particular, every arrow from $[\![A]\!]$ to $[\![B]\!]$ in $\mathcal{CONS}$ is the consumed form of a proof-module from A to B.

Proof. To construct the functor $[\![-]\!] : \mathcal{P}^h \longrightarrow \mathcal{CONS}$, we fix an interface $a_A : A \longrightarrow_v [\![A]\!]$ for every object A of $\mathcal{P}^h$, and associate to every morphism $\pi : A \longrightarrow_h B$ in $\mathcal{P}^h$ the (unique) consumed form $[\![\pi]\!] : [\![A]\!] \longrightarrow_h [\![B]\!]$ inducing a cell α in $\mathcal{P}$, with edges:

$$\begin{tikzcd} A \arrow[r, "\pi"] \arrow[d, dotted, "a_A"'] \arrow[dr, phantom, "\alpha"] & B \arrow[d, dotted, "a_B"] \\ [\![A]\!] \arrow[r, "[\![\pi]\!]"'] & [\![B]\!] \end{tikzcd}$$

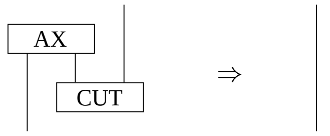
Functoriality of $[\![\pi]\!]$ is by construction, and fullness by reversibility of $\mathcal{P}$ (theorem 3). $\square$

Remark: theorem 5 follows from normalisation, confluence and reversibility, which are properties of the double category $\mathcal{P}$ abstracted away from their linear logic's origins. That way, semantics (= the consumed form) is reevaluated as the *difference* between a proof-module and its associated formula (= vertical semantics). This alters and completes our introductory principle (1). Indeed, in $\mathcal{P}$, the "vertical semantics" of a proof-net is the formula it proves, while its "consumed semantics" is what remains after complete exhaustion of the proof-net.

Remark: let us mention the existence of another double category modelling MLL, and enjoying modularity, not constructed in this article. There, the "vertical semantics" of a proof-net is its cut-free normal form, while its "consumed semantics" retains nothing of it: only ashes, the empty proof-net. In this alternative construction, the vertical semantics of a proof-net mirrors all its interactive content (like a Böhm-tree in the λ-calculus) and our introductory principle (1) is entirely respected.

4.4. *Axiom and cut links in the model*

By following (Lafont 1995) instead of (Girard 1987), that is keeping axiom and cut links implicit, we depart from the original and more usual syntax of multiplicative linear logic. Would it complicate matters to integrate the cut and axiom links in a double categorical treatment of linear logic? No, not really. One reason is that there exist simple translations between the two presentations. Every Girard net L-translates to a Lafont net by replacing every cut or axiom link with a wire. Conversely, every Lafont net G-translates to a Girard net by replacing every wire between two auxiliary ports by an axiom link, and between two principal ports by a cut link. Another reason is that cut-elimination of axiom and cut links is processed in Girard nets according to the following rule:



(12)

We take the opportunity to describe the double category $\mathfrak{D}_{\mathrm{G}}$ associated to Girard nets. Crucially, the rule hereabove (implicit in $\mathfrak{D}$) is mirrored by a cell in the double category $\mathfrak{D}_{\mathrm{G}}$. Modularity implies that this cell factors out as a bunch of new cells, for example the cell which "emits" a cut link of π pointing to a free port, and replaces it with a wire. Observe also that horizontal composition in $\mathfrak{D}_{\mathrm{G}}$ should not connect two premises or two conclusions[‡‡]. This motivates to equip every free port with an *orientation*, as follows. We define an object of $\mathfrak{D}_{\mathrm{G}}$ as a quadruple $(m, E_\Delta, F_\Delta, \mathtt{orient}_\Delta)$ where (m, E_Δ, F_Δ) is an object of $\mathfrak{D}$, and $\mathtt{orient}_\Delta : [m] \longrightarrow \{+, -\}$ is a map indicating whether $i \in [m]$ should be considered in Δ as a premise ($\mathtt{orient}_\Delta(i) = -$) or as a conclusion ($\mathtt{orient}_\Delta(i) = +$). Horizontal arrows from Δ to $\Gamma = (n, E_\Gamma, F_\Gamma, \mathtt{orient}_\Gamma)$ in $\mathfrak{D}_{\mathrm{G}}$ are the Girard nets:

1. which L-translate to a module $(m, E_\Delta, F_\Delta) \longrightarrow (n, E_\Gamma, F_\Gamma)$ in $\mathfrak{D}^h$,
2. where every premise of a link is connected to the conclusion of another link, and every conclusion to a premise. Here, we consider the ports x_i and y_j as premises when $\mathtt{orient}_\Delta(i) = +$ or $\mathtt{orient}_\Gamma(j) = -$, and as conclusion when $\mathtt{orient}_\Delta(i) = -$ or $\mathtt{orient}_\Gamma(j) = +$.

The horizontal category is self-dual with the endofunctor $(-)^{op}$ defined on objects as $\Delta^{op} = (m, F_\Delta, E_\Delta, -\mathtt{orient}_\Delta)$. A vertical arrow $a : \Delta \longrightarrow_v \Gamma$ in $\mathfrak{D}_{\mathrm{G}}$ is defined as a *normal* Girard net which L-translates to a vertical arrow ($[m, n]$-forest) in $\mathfrak{D}$. Here, normal means that a does not contain any cut-axiom pattern like (12). A cell in $\mathfrak{D}_{\mathrm{G}}$ is a 5-uple $(\pi, \pi', a, b, \mathfrak{P})$ where $\mathfrak{P}$ pairs tensor links with par links in $a^{\circledast}; \pi; b$ and verifies:

1. that every virtual link is paired,
2. that π' is the result of cut-eliminating all the pairs in $\mathfrak{P}$, plus some cut-axiom patterns.

Here, $a^{\circledast}$ is the normal net defined as in section 2.4 except for the cut and axiom links added/removed at its free ports to adapt the orientations and ensure that $a^{\circledast}$ is an horizontal arrow $a^{\circledast} : \Gamma \longrightarrow_h \Delta$ in $\mathfrak{D}_{\mathrm{G}}$. Cell composition and cell identities are defined as in $\mathfrak{D}$. The construction of $\mathfrak{D}_{\mathrm{G}}$ ends with the verification of the coherence properties. Thus, except for the orientation issue, the double categories $\mathfrak{D}$ and $\mathfrak{D}_{\mathrm{G}}$ are similar.

5. Conclusion

The theory of computation is mainly concerned with two manipulations:

1. to compute a program,

[‡‡] Remark.— Auxiliary and principal ports of a Girard net are called premise and conclusion in the sequel. For instance, a cut link has two premises and no conclusion, an axiom link has two conclusions and no premise.

2. to divide a program into subprograms for separate analysis and development.

With double categorical models (called tile model by Gadducci and Montanari) it becomes possible to divide not only the program (the static syntax = horizontal arrows) but also the computation (the dynamic syntax = cells). *Syntax* and *semantics* reunite in this modular perspective, as two aspects of the same phenomenon: *computation*.

Here, we apply the methodology to multiplicative linear logic, and reconstruct the meaning of a proof-net π of type A as what remains of π after complete vertical communication of A. Expressed this way, semantics appears as the *difference* between a term π and its type A. This differential principle, or the more interactive principle (1) exposed in introduction, should extend to various computation paradigms, from term rewriting to process calculi, leading hopefully to the spreading of a trinity enjoyed today by (multiplicative) linear logic: a static denotation, a dynamic denotation, and a modular syntax.

Acknowledgements

This article is the finish of a four-year stroll which started at INRIA in 1993 (Melliès 1994) in order to explain "syntactically" and repair the well-known non-congruence property of weak bisimulation in CCS and π-calculus, see (Melliès Bourgeois-Gironde 1997). At the time, fascinating connections with the usual Böhm tree semantics of λ-calculus, and similar "Böhm graph" semantics for Lafont interaction nets, were unshadowed. The line of research rebounced a few years later with the discovery at a CONFER workshop in München of the independent but strongly related work of (Gadducci Montanari 1996) on double categories, and finalized in Edinburgh with the analysis of the geometry of interaction and game semantics. I would like to thank Andrea Asperti, Pierre-Louis Curien and Jean-Jacques Lévy for their support at the very first stage of this work. I am also endebted for discussions and comments to Andrew Barber, Richard Blute and Hugo Herbelin. Finally, I would like to thank the anonymous referee for suggesting essential improvements in the presentation of this work.

References

S. Abramsky, P.-A. Melliès. Concurrent games and full completeness. In *Proceedings LICS'99*, Trento, 1999.

M. Barr. $*$-Autonomous Categories and Linear Logic. In *Mathematical Structures in Computer Science*, 1991.

G. M. Bierman. What is a categorical model of Intuitionistic Linear Logic? In *Proceedings TLCA'95*, Edinburgh, 1995.

R. Blute. Linear logic, coherence and dinaturality. In *Theoretical Computer Science*, 115:3-41, 1993.

R. Bruni and U. Montanari. Cartesian Closed Double Categories, their Lambda-Notation, and the Pi-Calculus. In *Proc. 14th Symposium on Logic in Computer Science*, Trento, July 2-5, 1999, IEEE Computer Society, 1999, pp.246-265.

J. R. B. Cockett, R. A. G. Seely. Weakly distributive categories. In *Journal of Pure and Applied Algebra*, 114:133-173, 1997.

V. Danos, L. Regnier. The structure of multiplicatives. In *Archive for Mathematical Logic*, 28:181-203, 1989.

A. Fleury, C. Rétoré. The mix rule. In *Mathematical Structures in Computer Science*, volume 4, number 2, June 1994.

F. Gadducci, U. Montanari. Tiles, rewriting rules and CCS. In J. Meseguer editor, *Rewriting logic and applications*, Electronic Notes in Computer Science, vol. 4, 1996.

N. Ghani. *Adjoint Rewriting*. PhD thesis, University of Edinburgh, 1995.

J.-Y. Girard. Linear Logic. In *Theoretical Computer Science* 50, pp 1-102, 1987a.

J.-Y. Girard, Multiplicatives. In *Rencontri del Seminario Matematico dell'Università e Policlinico di Torino*, special issue on Logic and Computer Science, pp 11-33, 1987b.

G. Gonthier, J.-J. Lévy, P.-A. Melliès. An abstract standardisation theorem. In *Proceedings LICS'92*, Santa Cruz, 1992.

G. Huet, J.-J. Lévy. *Call by Need Computations in Non-Ambiguous Linear Term Rewriting Systems*. Rapport de recherche INRIA 359, 1979.

B. Jacobs. *Categorical type theory*. Proefschrift aan de Katholieke Universiteit Nijmegen, 1991.

C.B. Jay. Modelling reduction in confluent categories. In *Applications of categories to Computer Science*, volume 177. London Mathematical Society Lecture Note Series, 1992.

J.W. Klop. Term Rewriting Systems. In *Handbook of Logic in Computer Science*, volume I, editors S. Abramsky, D. Gabbay, and T. Maibaum, Oxford University Press, 1992.

G. M. Kelly, M. La Plaza. Coherence for compact closed categories. In *J. Pure and Applied Algebra* 19, pp. 193–213, 1980.

G. M. Kelly, S. Mac Lane. Coherence in closed categories. In *J. Pure and Applied Algebra* 1, pp. 97–140, 1971.

Y. Lafont. From proof-nets to interaction nets. In *Advances in Linear Logic*, edited by Jean-Yves Girard, Yves Lafont and Laurent Regnier, LMS 222, Cambridge University Press, 1995.

J. Lambek. *Deductive systems and categories II*. Lecture Notes in Mathematics 87, Springer Verlag, 1969.

J. Lambek, P.J. Scott. *Introduction to higher order categorical logic*. Cambridge studies in Advanced mathematics 7, 1986.

J.-J. Lévy. *Réductions correctes et optimales dans le λ-calcul*. Thèse de Doctorat d'Etat, Université Paris VII, 1978.

R. Loader. Linear Logic, Totality and Full Completeness. In *Proceedings LICS'94*, 1994.

S. Mac Lane. *Categories for the working mathematician*. Springer Verlag, 1971.

P.-A. Melliès. *Non mobile models of mobility*. Manuscript, 1994.

P.-A. Melliès. *Description abstraite des systèmes de réécriture*. Thèse de l'Université Paris VII, December 1996.

P.-A. Melliès, S. Bourgeois-Gironde. *Three approaches to congruence*. Wittgenstein colloquium, Kirchberg am Wechsel, 1997.

A. J. Power. An abstract formulation for rewrite systems. In *Category Theory and Computer Science*, LNCS 389, Springer Verlag, 1989.

D. E. Rydeheard, J. G. Stell. Foundations of equational deduction: A categorical treatment of equational proofs and unification algorithms. In *Category Theory and Computer Science*, LNCS 283, Springer Verlag, 1987.

R.A.G. Seely. Modelling computations: a 2-categorical framework. In *Proceedings LICS'87*, 1987.

R.A.G. Seely. Linear logic, $*$-autonomous categories and cofree algebras. In *Conference on Categories in Computer Science and Logic*, volume 92 of *AMS Contemporary Mathematics*, pages 371-382, June 1989.