Arch. Math. Logic (1989) 28:181–203

Archive for
Mathematical
Logic
© Springer-Verlag 1989

The Structure of Multiplicatives

Vincent Danos and Laurent Regnier

Equipe de Logique, CNRS UA 753, UFR de Mathématiques et d'Informatique de Paris VII, Tour 45–55, 2, place Jussien, F-75251 Paris Cedex 05, France

Abstract. Investigating Girard's new propositionnal calculus which aims at a large scale study of computation, we stumble quickly on that question: What is a multiplicative connective? We give here a detailed answer together with our motivations and expectations.

Introduction

The questions raised here stem from Girard's construction of a new logic, *linear logic*. There the usual logic gets split in three parts, *multiplicative*, *additive*, and *exponential*. As expected from a good decomposition, each part has distinct properties, the combination of which gives anew the usual logic without loss of expressive power. We content here ourselves with the first one, and try to inspect its properties.

So we want to know: *what's a multiplicative connective?* But answering this question leads quickly to boring syntactical considerations if we don't bear in mind why we're interested in it. Let us begin with a list of problems we expect to make easier. Some internal problems of linear logic theory:

– We wish to have a better understanding of the correctness of proof-nets (a new way to denote linear proofs). This could allow us to: analyze the trips used to prove correctness, and find a more feasible way to do it (in P-time), or even to extend correctness conditions to a larger part of linear logic.

– We also want to discuss the first definition of the multiplicatives given by Girard because: it makes use of heavy objects, some of them seeming unnecessary, and has a quite unsatisfying coherent semantics.

– Last, we would like to relate precisely the process of verifying correctness and the execution of a proof-net. That is: what do we really ask, when we ask a multiplicative program/proof to be sequential.

Some external problems, which are less formal:

Offprint requests to: V. Danos

– On one hand, we believe linear logic to deal with time in computation, that is of real dependencies between informations processed during computation. In other words linear logic should be a communication description system (we even hope it to be the first step in designing logical parallelism). Indeed, the two basic multiplicatives, *times* and *par* clearly behave as correlation descriptors between communication channels in a system without synchronization. A complete development of this intuition leads to a new evaluation mechanism which is parallel in a restricted sense. Yet this idea still lacks of theoretical support, and we'll try elsewhere to build on it. With this intuition our question becomes: *What is the structure of these multiplicative descriptors?* We want to catch every object living in the simple multiplicative world. This is important, as linear logic won't describe all kinds of parallelism. Eventually, the question is: *How much multiplicative parallelism can we catch?*

– On the other hand, we believe linear logic to deal with space in computation, that is of memory allocation. Hence we can think of the multiplicatives as descriptions of packed informations, which will be processed together. However, we didn't further investigate this idea.

– Last, we would like to study the problem of modularity for the multiplicative programs, but postpone the discussion that this problem deserves.

These are the basic motivations for our study. Some of the questions we pointed at are solved, but not all of them! We thought it a good idea to begin the study in the sequent calculus formulation of multiplicative linear logic, partly because it's known as a simple formalism, and partly because we are interested at a theoretical level in comparing the two formulations of linear logic: the sequential one, **LLM** for multiplicative linear logic, and the most promising one, proof-nets, $\mathbf{PN_0}$ for short. We therefore start by a rather sketchy presentation of both, for a minimum of self-containedness. In the next two sections we study the multiplicatives respectively in sequential syntax, and proof-nets. Then we proceed to a comparison between the results in both syntaxes. The last section being devoted to the feedback of the preceding to the existing theory.

1. Linear Multiplicative Calculus

We give here the two known syntaxes denoting multiplicative proofs and show how they relate. An extensive presentation is given in [Gir 87], together with the missing proofs.

1.1. Sequential Presentation

This syntax follows Gentzen's sequent calculus for classical logic, traditionally called **LK** (see [Gentzen 69]). Negation is restricted to atomic formulas, hence there is no rule for it. Therefore we work modulo de Morgan identities:

$$A^{\perp\perp}=A, \quad (A\otimes B)^{\perp}=A^{\perp}\wp B^{\perp}, \quad (A\wp B)^{\perp}=A^{\perp}\otimes B^{\perp}.$$

Hence we no more need left rules, and handle them with the dual connective. This is an usual trick for shortening the list of rules. The formulas individuated in each of the following rules are called principal formulas, and the remaining formulas are called the context(s).

$$\frac{\text{axiom}}{\vdash A^{\perp}, A}, \qquad \overset{\text{cut rule}}{\frac{\vdash \Gamma, A \quad \vdash \Delta, A^{\perp}}{\vdash \Gamma, \Delta}},$$

$$\overset{\text{exchange rule}}{\frac{\vdash \Gamma, A, B, \Delta}{\vdash \Gamma, B, A, \Delta}},$$

$$\overset{\text{times rule}}{\frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}}, \qquad \overset{\text{par rule}}{\frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \wp B}}.$$

This is nothing but **LK** where we rejected the two structural rules of weakening and contraction.

$$\overset{\text{weakening rule}}{\frac{\vdash \Gamma}{\vdash \Gamma, A}}, \qquad \overset{\text{contraction rule}}{\frac{\vdash \Gamma, A, A}{\vdash \Gamma, A}}.$$

The main property of **LK** which still holds for **LLM** was proved by Gentzen.

Theorem 1. *The cut rule is redundant in both* **LK** *and* **LLM**.

This means that one can always avoid in a proof the use of formulas which do not appear in the conclusion, as long as there are no non-logical axioms. This theorem which is constructive yields a kind of logical calculus over proofs. Take a proof involving general principles which are used in special cases and, following the process described by Gentzen, reduce it to a *normal* form where these lemmas are no more used. It then seems a good idea to interpret this normalization as an abstract model of computation. Yet the process of normalization is in **LK** non-deterministic, and makes this idea not so good. This is why classical logic is non-constructive. On the contrary the process is deterministic in the subcase of **LLM** (and even for the complete linear logic), making it a constructive logic, despite the presence of an involutive negation. This founds the claim of linear logic to be a model of computation.

We could have chosen an another rule for conjunction:

$$\overset{\text{with rule}}{\frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}}.$$

These two formulations are equivalent modulo the structural rules we just refused to keep in the system: *times* + contractions simulates *with*, and *with* + weakenings simulates *times*[1]. Without them, we now have two distinct conjunctions *times* and *with* which are no more variant. The former requires nothing about the contexts and belongs to the multiplicative fragment, while the latter

[1] One has no control over the number of contractions or weakenings required for the simulation

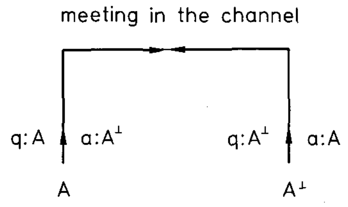


Fig. 1. Typed information in a basic channel

requires both contexts to be equal and belongs to the additive fragments. The same remarks apply to the classical disjunction which get also duplicated.

We shall now see how fruitful this decomposition is.

1.2. Proof-Nets

We shall try here to present proof-nets in an other way than the original one (see [Gir 87] or [Gir 87b]). The following presentation is intended to support a view of proof-nets as typed communication systems, which we already mentioned in the introduction. This will allow us in the sequel to set discussions at a more intuitive level.

So, let us start from scratch, with a question. We would like to build a system of information transfer where:

– the information is moving,

– the elementary transfer is the meeting of two informations.

We don't say anything about the kind of informations involved. But some of the informations are answers (A), while the others are questions (Q). Hence a meeting is simply the answering of a question. We give us elementary channels where informations can travel in both directions. Meetings occur when two informations traveling in opposite directions are in the same channel.

Obviously some local requirements should be stated to ensure that meetings are meaningful. First we type our channels, to prevent awkward meetings such as: what time is it?/red. Then, we also want to avoid: what time is it?/what time is it?, or even worse: red/blue. But we don't want to explicitly state which informations are questions, and which are not. In other words we want to ensure correct meetings without orientation of the channels. Therefore we introduce the central connective of $\mathbf{PN}_0$, that is the linear negation. A question of type X is a answer of type $X^\perp$ and reciprocally.

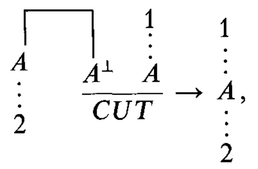
$$Q:X \equiv A:X^\perp, \qquad A:X \equiv Q:X^\perp.$$

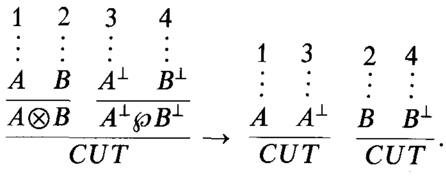
This provides a symmetric formulation. We sum up this in Fig. 1. Basic channels are axioms. Now we allow basic channels to be connected by mean of a cut (terminology is a bit confusing here, for the cut is a plugging!) with the restriction that only dual atoms can be connected, for the same reasons of compatibility of meetings. We call channels these connections of basic ones. Yet there is another condition about cuts, that we do not close a channel on itself. This would be a system without any outputs, nor any possibility of inputs.

We try then to handle more complex structures, i.e., with more than one channel. To do this we a priori need a single connective, the meaning of which should be the grouping of several channel extremities in one. Then we extend cuts between arbitrary formulas built with our connective, checking that there is the same number of atoms on each side and that they are pairwise dual. Yet this does not suffice to prevent formation of closed channels. What is the matter? A single connective does not carry enough information about the channels it gathers to hope that we can state a general condition over this structures preventing them to be closed by an undesirable cut. We must have some local way to describe dependencies between the channels we are combining. We try again with *two* binary connectives. Suppose we want to group two atomic extremities in one, there are only two cases, whether they are not belonging to the same one and never will, in which case we record this by using a *times* for the gathering, or they are or will be, in which case we use a *par*. This time it works, i.e. we know how to characterize a subset of these structures where any locally correct cut will keep the channels opened.

Let us add a few comments. The difference between cuts and axioms is that, on one hand cuts are possible between a set of atoms, while axioms remain atomic, and on the other hand cuts remain passive connections, whereas operations respecting the Q/A structure are possible in axioms. This is, by the way a sound extension of this system. Other extensions should be reachable by including the remaining of linear logic, leading probably to more elaborate meetings involving unification. Moreover such extensions are badly needed as the multiplicative calculus is quite poor[2].

Computation in this system, if we follow our intuitive model is simply the launching at every channel extremity of a question with appropriate type (hence this can be a positive information), and the resulting meetings of informations. How much one can expect of this approach for the design of a logical system for parallelism, is still unclear. However there is a more classical way to present computations in proof-nets, by showing how one get rid of the cuts.





All reductions are ended in linear time in the size of the proof-net, because the computation always shortens it. This shows again the limits of this part of linear

[2] It encodes only linear λ-calculus with conjunction

logic. Only in this syntax, is the calculus really deterministic, as we shall next see by giving the translation between both syntaxes.

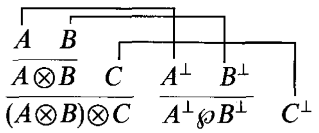
1.3. Translation of Sequential Proofs into Proof-nets

Let us see how the translation works. To any sequential proof Π we associate a proof-net N, with the same conclusions.

$$\begin{array}{cc}
\overline{\vdash A^{\perp}, A} & \overline{A \quad A^{\perp}} \\[2ex]
\dfrac{\begin{array}{cc}\Pi_1 & \Pi_2 \\ \vdots & \vdots \\ \vdash \Gamma, A & \vdash \Delta, A^{\perp}\end{array}}{\vdash \Gamma, \Delta} & \begin{array}{c}\Gamma \textcircled{N_1} \quad \textcircled{N_2} \Delta \\ \dfrac{A \quad A^{\perp}}{CUT}\end{array} \\[6ex]
\dfrac{\begin{array}{cc}\Pi_1 & \Pi_2 \\ \vdots & \vdots \\ \vdash \Gamma, A & \vdash \Delta, B\end{array}}{\vdash \Gamma, \Delta, A \otimes B} & \begin{array}{c}\Gamma \textcircled{N_1} \quad \textcircled{N_2} \Delta \\ \dfrac{A \quad B}{A \otimes B}\end{array} \\[6ex]
\dfrac{\begin{array}{c}\Pi \\ \vdots \\ \vdash \Gamma, A, B\end{array}}{\vdash \Gamma, A \wp B} & \begin{array}{c}\Gamma \quad \textcircled{N} \\ \dfrac{A \quad B}{A \wp B}\end{array}.
\end{array}$$

As the notation is graphical, the exchange rule doesn't induce any transformation on the proof-net, yet if we were to make a notation for a machine we should denote this rule. Note that in **LK** the information conveyed by the names of the two multiplicative connectives is redundant, as the rules themselves show which connective we are using, while in $\mathbf{PN}_0$ this information is needed. Hence the description of a proof is stricter in $\mathbf{PN}_0$, and we stand closer in this syntax to the real objects we have in mind. Anyway the sequential presentation seems clearly more expensive and cumbersome. And indeed, the translation is far from being injective. What does it mean? Let us have a look at an example, which we borrow from [Gir 87]. Both the following proofs translate in the same proof-net.

$$\dfrac{\dfrac{\dfrac{\vdash A, A^{\perp} \quad \vdash B, B^{\perp}}{\vdash (A \otimes B), B^{\perp}, A^{\perp}}}{\vdash (A^{\perp} \wp B^{\perp}), (A \otimes B)} \quad \vdash C, C^{\perp}}{\vdash ((A \otimes B) \otimes C), (A^{\perp} \wp B^{\perp}), C^{\perp}} \qquad \dfrac{\dfrac{\dfrac{\vdash A, A^{\perp} \quad \vdash B, B^{\perp}}{\vdash A \otimes B, A^{\perp}, B^{\perp}} \quad \vdash C, C^{\perp}}{\vdash (A \otimes B) \otimes C, A^{\perp}, B^{\perp}, C^{\perp}}}{\vdash (A \otimes B) \otimes C, A^{\perp} \wp B^{\perp}, C^{\perp}}$$



Now, we claim that this situation is general. Any two sequential proofs which have the same meaning become identical through this translation (completeness) while sequential proofs which have not the same meaning translate into distinct proof-structures (consistency). Furthermore, there is a formal semantic of linear proofs, coherence semantics [Gir87] which makes this statement precise. Hence $\mathbf{PN_0}$ is optimal.

This is reflected in the way computations differ in both syntaxes. One is local and with no syntactic loss of time, while the other is constantly reordering the rules to achieve its goal.

A proof-net is simply a correct proof-structure, i.e., belonging to the codomain of this translation. But there is an intrinsic notion of correctness for proof-structures.

1.4. Proof-nets Correctness – Sequentialization

For stating the correctness condition, we have before to associate *traveling instructions* to each link, as well as to conclusions:

axiom: If you move upward through an atom, hence entering an axiom link then exit it by the dual atom, now moving downward.

conclusion: If you enter a conclusion moving downward, then rebounce, now moving upward.

par: If you enter a par link moving downward just leaving the left (resp. right) premise and it's switched to the left (resp. right) then keep on moving downward. On the contrary if you enter the par through a premise and the link is switched to the opposite side, then rebounce as in a conclusion. Last, if you enter the par from its conclusion then keep on moving upward choosing as next formula the left (resp. right) premise if the switch is left (resp. right).

times: If you enter a times link moving downward just leaving the left (resp. right) premise and it's switched to the left (resp. right) then keep on moving downward. On the contrary if you enter the par through a premise and the link is switched to the opposite side, then move upward through the other premise. Last, if you enter the times from its conclusion then keep on moving upward choosing as next formula the left (resp. right) premise if the switch is right (resp. left).

The traveling instructions are deterministic up to the choice of a switch for each par and times link, left or right.

Definition 1. Given a switch in a proof-structure, a trip is a cyclic sequence of moves respecting the traveling instructions associated to the switch.

Either a trip visits all formulas twice, in which case it's a *long* one, or it does not, and we call it a *short* one.

Theorem 2. *A proof-structure is correct, i.e., sequentializable, iff all its possible trips are long ones.*

Proof (see [Gir 87]). How ensuring correctness of proof-nets without an intrinsic condition? There are three main cases: if the structure was given by the above translation, or if it came from the normalization of a correct one, or if it was built by a sequential application of the connectives, satisfying the dependencies we said that they describe, then it is a correct one. So one could argue that the intrinsic condition we just showed is of no practical interest, as we can ensure correction by a lot of other means. Then, wherefore a direct correctness criterion? This allows direct non-sequential writing of a linear proof, but also a more geometrical treatment of proofs by preventing the steady recourse to some sequential representation of the proof-nets. Last, modules would be less general, because without such a condition one would have to define them through the sequent calculus.

Well, but the correctness condition we just defined is exponential in the size of the proof-structure being checked. This denied any practical interest to the questions we mentioned, as long as there was no feasible correctness criterion. Yet, following some ideas of this study the first author found a feasible one, which is even adequate to a larger fragment[3] of linear logic [Da 88].

2. Sequential Multiplicatives

2.1. Geometry of a Multiplicative Rule

We want to define the notion of general multiplicative rule. The rules of **LLM** all share the property of being *unconditional* about the context, and *conservative* over the atoms[4]. By unconditional, we mean that they do not require anything about the contexts, whereas by conservative we mean that the premise(s) of the rule has exactly the same atoms as its conclusion. Hence, a basic multiplicative rule is completely characterized by the organization of its principal formulas. We obviously should define general rules in such a way that this fundamental property still holds. Therefore a general rule for the n-ary connective C should look this way:

$$\frac{\vdash \Gamma_1, A_1, \dots, A_{i_1} \quad \dots \quad \vdash \Gamma_p, A_p, \dots, A_{i_p}}{\vdash \Gamma_1, \dots, \Gamma_p,\ C(A_1, \dots, A_n)}\,.$$

Let us now drop the contexts as any rule will operate on them by a mere concatenation. The rule becomes with this simplification:

$$\text{general multiplicative rule}$$
$$\frac{\vdash A_1, \dots, A_{i_1} \quad \dots \quad \vdash A_p, \dots, A_{i_p}}{\vdash C(A_1, \dots, A_n)}\,.$$

We shall describe this rule with a *partition* defined over its principal formulas. Let R be a n-ary rule, the corresponding partition p_R is defined over $\{1, \dots, n\}$ as follows: i and j belong to the same class of p_R if the formulas A_i and A_j belong to the same premise of R. Conversely, given a partition p we can check if a rule satisfy it. Renumber the principal formulas from left to right in the conclusion, and ask if the induced partition is really p.

[3] Including a variant of the weakening rule
[4] In fact the rules of **LLM** are the only rules of **LK** satisfying this

For instance, the following rules:

$$\frac{\vdash A, B \quad \vdash C}{\vdash D(A,B,C)}, \quad \frac{\vdash A, B, C}{\vdash \mathscr{D}'(A,B,C)}$$

are respectively represented by the partitions $\{\{1,2\},\{3\}\}$, and $\{\{1,2,3\}\}$ while the basic multiplicatives *times* and *par* are respectively represented by $\{\{1\},\{2\}\}$ and $\{\{1,2\}\}$.

Choosing partitions as rules amounts to say that we disregard both the order of the premises, and the order in the premises. The external order is irrelevant because one can always switch it, hence there is clearly no gain of expressive power with considering partition + an ordering of the classes, furthermore this would make the calculus redundant. The internal order we also disregarded, because we don't want to have both this rules:

$$\frac{\vdash A, B}{\vdash C(A,B)}, \quad \frac{\vdash B, A}{\vdash C'(A,B)}$$

for the second can be simulated by the first (which is the par rule) together with one application of the exchange rule, which is precisely designed for this purpose. Exchange is an operationally very important rule, that dictates which atom will be connected to which in a complex connection (other name for the cut rule!). Multiplicatives, on the contrary have no operational meaning, they are only correlations descriptors which ensure a good execution or communication without playing part in it. As this is a common feature of type systems, we don't need to say more. So general multiplicatives shouldn't hide exchanges, because we don't want our generalization to lose this static aspect of multiplicatives. This should make clear at a technical level why we use partitions.

Let us discuss this at a more intuitive level, following our basic intuition of the multiplicatives. As they describe dependencies between communication channels, the refusal of a internal order expresses the symmetry of the relation "to be linked to". But we could think of this dependencies as antisymmetrical ones, interpreting them as temporal. This interpretation forces the rejecting of the exchange rule, and leads to non-commutative linear logic of which an account is given in [Gir 88], and which is *un autre monde.*

2.2. Definition of the Sequential Connectives

But one rule is not enough to define a connective. A sequential connective C will involve a set of partitions representing the right rules, and as a consequence of our simplification at the start, the left rules will be handled by the dual connective C^*. Note that the dual of C, C^* is included in the definition of C, and dually the dual of the dual, that is C itself is included etc. That's duality. To be totally pedantic, we mean that if we denote by O_C^* the set of left rules for C, then $O_C^* = O_{C^*}$, i.e. they are the right rules of C^*. Notation O_C is designed to remind that each of these partitions describe an *organization* of the premises which allow to derive C.

Yet this is incomplete, because introducing the new constructs C and C^*, we said no word about how we intended to extend the notion of computation to them, that is how we extend cut-elimination. There are rules as non-logical axioms, with which one cannot eliminate cuts, and this means that such computations do not belong to the system and should be made outside. Here, this is not the case. For elimination of cuts on C, we have to substitute to this configuration:

$$\frac{\dfrac{\vdash A_1,\ldots,A_{i_1}\ \ldots\ \vdash A_p,\ldots,A_{i_p}}{\vdash C(A_1,\ldots,A_n)} \qquad \dfrac{\vdash A_1^{\perp},\ldots,A_{j_1}^{\perp}\ \ldots\ \vdash A_q^{\perp},\ldots,A_{j_q}^{\perp}}{\vdash C^*(A_1^{\perp},\ldots,A_n^{\perp})}}{\vdash}$$

where C and C^* where introduced by the rules p and p^*, a proof of the same, here the empty sequent for simplification, by means of cuts between each couple of dual sub-formulas A_i and $A_i^{\perp}$, and exchanges. This is usually called the *key-step* in normalization. For instance the basic case is handled through the transformation:

$$\frac{\dfrac{\vdash A \quad \vdash B}{\vdash A\otimes B} \qquad \dfrac{\vdash A^{\perp},B^{\perp}}{\vdash A^{\perp}\wp B^{\perp}}}{\vdash} \quad\longrightarrow\quad \frac{\dfrac{\vdash A \quad \vdash A^{\perp},B^{\perp}}{\vdash B^{\perp}} \qquad \vdash B}{\vdash}$$

but this won't work for any two partitions p and p^*. Let us define $G(p,p^*)$ as the unoriented graph which has for vertices the classes of p and p^*, two of them being linked iff they share a point.

Definition 2. Two partitions are said to be orthogonal iff their induced graph is acyclic and connected.

For instance, the two rules we already showed (D and D') are not orthogonal, their graph being not acyclic, while the two basic multiplicatives obviously are. This is shown in Fig. 2.

Lemma 1. *A key-step between two partitions succeeds iff they are orthogonal.*

Proof. Let p and p^* be two partitions between which we try to perform the key-step. A cut between two dual sub-formulas corresponds to a retraction of $G(p,p^*)$ along the associated edge. (By retraction along an edge, we mean erasing it and identifying both extremities). More precisely, suppose you made any number of cuts, two sub-formulas are now in the same sequent iff their corresponding edges are adjacent in the retracted graph. A key-step succeeds if you manage to derive the empty sequent, that is to retract the graph in one point, with no more edges. A

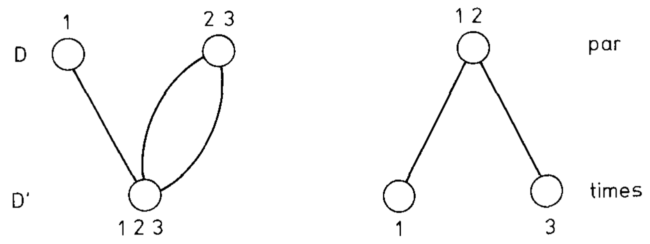


Fig. 2. Examples of induced graphs

graph admits such a retraction iff it is acyclic and connected as one can easily verify.

Note that the order of the cuts is indifferent.

If P and Q are sets of partitions, we write $P \perp Q$ if they are pointwise orthogonal. The Lemma 1 justifies the following definition:

Definition 3. A sequential multiplicative n-ary connective C consists of two sets of partitions O_C and O_C^* over $\{1,\ldots,n\}$ such that $O_C \perp O_C^*$.

A consequence of the definition is:

Proposition 1. *All the right (resp. left) rules of a n-ary connective have the same number of classes, say c (resp. c*) and* $c+c^*=n+1$.

Proof. An acyclic connected graph satisfies $v-e=1$, where v is the number of vertices and e the number of edges. □

Theorem 3. LLM + *sequential multiplicative connectives still admits cut-elimination.*

Proof. We only give a sketch of the proof, as it is only a minor variation of the classical proof of Gentzen for **LK**. In particular, we shall pay no attention to the exchange rules.

The main point is to prove that given a proof Π ending with a cut-rule with principal formula A:

$$\dfrac{\begin{matrix}\Pi_1 & \Pi_2\\ \vdots & \vdots \\ \vdash \Gamma, A & \vdash \Delta, A^\perp\end{matrix}}{\vdash \Gamma,\Delta}$$

one is able to produce a new proof of the same sequent only by means of cuts on subformulas of A. (In the case A is atomic, this means without any new cut).

There are two cases. Whether both of the last rules of Π_1 and Π_2 are rules introducing respectively A and $A^\perp$, in which case it is what we called a *key-step.* Or one of them is another rule, and we have to perform a *commutation step.* The former case has already been examined in Lemma 1, except for the case of an axiom, which is easy. For the latter, suppose that the last rule of Π_1 is not a rule introducing A. Then it can only be a general multiplicative rule (or a cut, but the step remains the same). Hence Π looks like this:

$$\dfrac{\dfrac{\begin{matrix}\vdots & & \vdots\\ \vdash A, B_1,\ldots,B_{i_1} & \cdots & \vdash B_p,\ldots,B_{i_p}\end{matrix}}{\vdash A, C(B_1,\ldots,B_n)} \qquad \begin{matrix}\Pi_2\\ \vdots\\ \vdash A^\perp\end{matrix}}{\vdash C(B_1,\ldots,B_n)}.$$

And we rewrite it into:

$$\dfrac{\dfrac{\begin{matrix} & \Pi_2 \\ \vdots & \vdots\\ \vdash A, B_1,\ldots,B_{i_1} & \vdash A^\perp\end{matrix}}{\vdash B_1,\ldots,B_{i_1}} \qquad \cdots \quad \begin{matrix}\vdots\\ \vdash B_p,\ldots,B_{i_p}\end{matrix}}{\vdash C(B_1,\ldots,B_n)}.$$

Elimination of the cut consists in performing commutation steps on the proof until the configuration is a key-step, where the cut can be split in smaller ones.

2.3. Decomposable Connectives – Packing Problem

So we constructed a sound extension of **LLM**. Some of the new connectives were already definable in **LLM**, the *decomposable* ones, that is any connective which is a combination of the basic ones, *times* and *par*. In other words, some dependencies are describable by the only means of the binary ones, i.e. no dependency, this is *times*, or dependency which is handled by the *par*.

Definition 4. A connective is decomposable iff there is a combination of *times* and *par*, having for rules the same two partitions sets.

For example, the connective D, half-defined by the right rule $\{\{1,2\},\{3\}\}$, we already looked at, is decomposable, if we complete the definition by choosing for left rules

$$\{\{1\},\{2,3\}\} \quad \text{and} \quad \{\{1,3\},\{2\}\},$$

for the combination $(\cdot \parr \cdot)\otimes\cdot$ has the same rules. We shall now add a few remarks.

First, not all the connectives are decomposable. So, we really increased the expressiveness in describing dependencies with the new connectives. For instance there is no binary combination who yields as only right (or left) rule $\{\{1,3\},\{2\}\}$. A more serious example is

$$O_C=\{\{\{1,3\},\{2,4\}\},\{\{1,2\},\{3,4\}\}\}$$

and

$$O_{C^*}=\{\{\{1,4\},\{2\},\{3\}\},\{\{1\},\{2,3\},\{4\}\}\}.$$

More serious because this connective does not admits any extension, for $O_C^\perp=O_{C^*}$ and $O_{C^*}^\perp=O_C$. (We write $P^\perp$ for the set of partitions that are orthogonal to all the elements of P).

Second, the connective D', with right rule $\{\{1,2,3\}\}$ and left one $\{\{1\},\{2\},\{3\}\}$, which is the ternary *par*, is twice decomposable! As well as $\cdot\parr(\cdot\parr)$, there is $(\cdot\parr\cdot)\parr\cdot$, decomposing it, and this means that partitions fix the slight syntactic default, that associativity is only provable and not a syntactical identity.

Last, there is the *packing problem*. Let X be a decomposable connective with right rules O_X. If we want to prove a sequent using X, we are faced with an alternative: whether we introduce X with one of its rules, or we build it stepwise, using its decomposition. Surprisingly, the second method is more powerful. There are proofs involving X which one cannot reach by the first alone. Let us look at an example:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash A,A^\perp \quad \vdash B,B^\perp}{\vdash (A\otimes B),B^\perp,A^\perp} \quad \dfrac{\vdash C,C^\perp \quad \vdash D,D^\perp}{\vdash (C\otimes D),D^\perp,C^\perp}}{\vdash (A^\perp\otimes C^\perp),B^\perp,D^\perp,(C\otimes D),(A\otimes B)}}{\vdash ((A^\perp\otimes C^\perp)\parr B^\perp),D^\perp,(C\otimes D),(A\otimes B)}}{\vdash (((A^\perp\otimes C^\perp)\parr B^\perp)\parr D^\perp),(C\otimes D),(A\otimes B)}}{\vdash ((A\otimes B)\parr(C\otimes D)),(((A^\perp\otimes C^\perp)\parr B^\perp)\parr D^\perp)}.$$

Note that the last rule could not be a partition of one of the conclusion connectives. Because any attempt would yield a premise with the other conclusion and some, but not all, of the principal formulas of the rule, and such sequents are never provable. For the same reasons, one cannot even prove $\vdash X, X^*$, i.e. the non-atomic axiom, with partitions! Obviously, the partitions of a decomposable connective do not describe all proofs involving it. They only describe the proofs where the construction of the connective is independent of the context. Precisely, any time there is a stepwise building of X, a decomposable connective, in a sequential proof, all the binary rules can be packed in one iff one can, by means of rules commutations, make them all appear consecutive. That's a striking difference with the syntax we will next study, proof-nets, where decomposable connectives behave exactly as their decomposition. General multiplicatives are more expressive in the non-sequential approach.

3. Multiplicatives in Proof-nets

We shall not directly define general multiplicatives in proof-nets, but introduce them through the problem of modularity.

3.1. A Dual Representation of Proof-nets

Let us twist a bit the usual presentation of proof-nets, which we gave in the first section. To a proof-structure we now associate a graph which has for edges its formulas, and for vertices the links and axioms. In the special case of conclusions, i.e., formulas which are not premises of any rule, we add a vertex. Note that conclusions behave like links in the definition of trips, so this isn't surprising. Anyway we have to add these vertices in order to have a real graph. All this is done for convenience and carries no new idea. Figure 3 exemplifies this slight transformation.

3.2. A New Correctness Criterion

To a proof-structure in graph-form, we associate a family of subgraphs. To obtain one of them, erase for each *par* vertex one of its premises, that is one of its upper

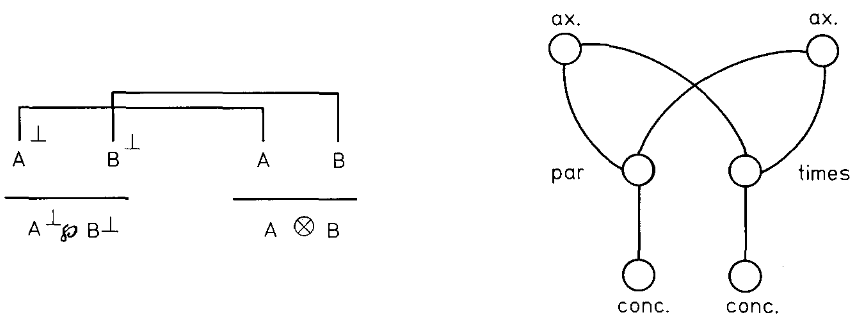


Fig. 3. Dual representation

edges. There are 2^p such graphs, where p is the number of *par* in the original graph. We recognize the traditional switching of a proof-net (and keep the name). Yet this time, the subgraphs corresponding to the trips, in a precise way we will later show, are less numerous.

Theorem 4. *A proof-structure is correct iff all its associated subgraphs are acyclic and connected.*

This criterion defines the same family of correct proof-nets as does the original one (see Theorem 2). We shall prove it in the last section.

3.3. Modules

We already know when (also in the sequential case) two correct proof-nets are connectable. This is when the two formulas being used in the connection are dual. Now the general problem is the following: given two arbitrary pieces of proof-structures which have not necessarily an autonomous meaning, how ensure that a general connection yields a correct result? There is an answer, and this suggests that we can define a notion of autonomy in this part of linear logic which by far exceeds our intuitive, semantic notion of module. As we know how to handle this general notion of module with respect to compositionnality, this allows non-semantic slicing of a multiplicative program. We can define a correct geography of a program, even if it can't be understood, for instance a spatial one. If this is possible in the multiplicative realm, it's because of the absence of global interaction during execution, and therefore we can reasonably hope to describe the behaviour of a module only by looking at its border. On the contrary, additives execution involves global interactions, and the notion of module is more problematic, at the present stage of theory.

Given a graph and a subset X of its vertices, we generate a subgraph in the following way: keep any edges internal to X, and for each edge linking X to its complementary, replace the vertex not in X by a new one, which will be called a *border vertex.*

Definition 5. A module is a subgraph generated as above by an arbitrary set of vertices chosen in a proof-structure in graph-form.

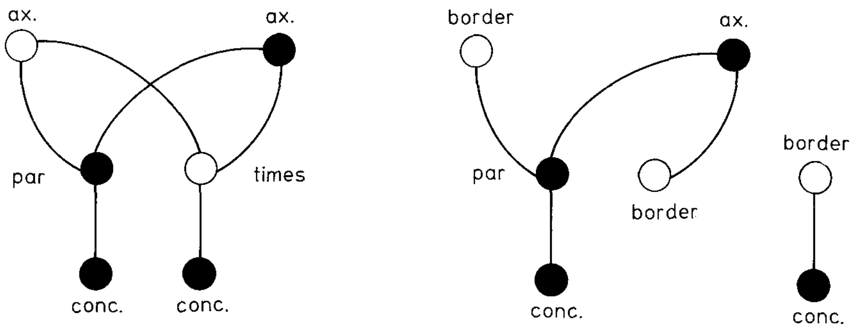


Fig. 4. A module cut in the preceding example

The *border* of a module is the set of border vertices or equivalently the set of links which lack either a premise or a conclusion. There are particular modules which are useful for reasoning about proof-nets, and which we shall list as examples.

First, the *tree* of a formula, say A, is a module, which we have already encountered in the first section. Take a proof of the non-atomic axiom $\vdash A, A^\perp$ and select for vertices all the links corresponding to the connectives A is built on. This is an other way to define a formula-tree. Their border edges (edges incident to a border vertex) are the atoms of the formula.

Second, take any proof-structure and select a maximal connected set of axioms, conclusions and *times* vertices, this yields a very special kind of modules which we call *blocks*. They have the remarkable property of being necessarily acyclic if the proof-structure is correct. Because they are never mutilated in any of the subgraphs used to check correction. Hence they behave as a simple vertex with respect to correction. Close examination of correctness in this frame leads to new and faster conditions, although equivalent, as explained in [Da 88]. The border edges of a block are either premises or conclusions of a par rule.

Last, we slice our last example (in Fig. 3) to show a concrete module in Fig. 4.

Surely, there are modules for which one should not even ask about composing them with any other, because they are already ill-formed. A correct module is a module for which no switching yields a cyclic graph. Note that we dropped the requirement of connection. We only consider correct modules in the following. We also ask that the border of a module is not void.

Now we turn to the task of describing the behaviour of a module on its border. Intuitively we should describe the potential connections between border vertices. A switching of the module defines a partition on the border: two border vertices will be in the same class iff they are connected in the switched subgraph.

Definition 6. The type of a module is the set of partitions induced by all the possible switchings on its border.

For instance blocks have a simple behaviour on their border, for their type only contains the trivial partition with one class. We choose to call it a type following [Gir 87b], for the function of a type is indeed in general to carry some compact information about an object, sufficient to secure compositions with the remaining world, or assert more elaborate properties of it. For connecting two modules, check that they have the same border, then pairwise identify the border vertices, and erase them. This connection is correct iff the resulting structure is a proof-net.

Theorem 5. *Two modules are connectable iff their respective types are orthogonal.*

Proof. This is exactly the same notion of orthogonality that we defined in the first part of the study. Let switch the resulting structure, this defines a switching on each module, and also two partitions of the border, say p_1 and p_2. As the modules are correct, there will be no internal cycles. Now retract in the switched graph the internal edges, i.e., edges which belonged to the modules. The result is nothing but $G(p_1, p_2)$. It's easy then to verify that a retraction of any graph is acyclic and connected iff it's already true for the original graph.

It does make sense to liberalize connections to the case of modules having distinct borders with a non-empty intersection. The connection then is correct iff the resulting structure is still a (correct) module, its border being now the symmetrical difference of the initial ones. There is another softening of the requirements which is more problematic: that we shall accept modules with void borders. We are tempted to say that such a module is exactly a proof-net, but how are we to interpret connections of two proof-nets through a void border? The natural way is to accept this as a correct proof, because two void types vacuously satisfy the plugging condition. Yet such proof-nets are not sequentializable. This can be fixed by adding in the sequential calculus the mix rule:

$$\frac{\vdash \Gamma \quad \vdash \Delta}{\vdash \Gamma, \Delta}$$

and this is indeed a sound extension for which there is a good generalization of correctness; see [Da88]. Yet the pertinence of this new rule, which is connected to the weakening rule, is discussable as we still do not understand its algorithmic content, allowing parts of a program not to communicate at all.

Now we proceed to a reformulation of this in a logical way, and introduce general multiplicative connectives.

3.4. Generalized Connectives

We use the Howard-isomorphism between programs and proofs backwardly, to reach our goal, that is a definition of general multiplicatives in proof-nets. As in the sequential case, we define one by two sets of partitions over the same domain, which are pointwise orthogonal. Yet there is a slight change. In the sequential case it seemed natural to define a connective by the rules that introduce it, that is to describe the external situations allowing its derivation. Here the connective is rather defined by its internal reaction to the general situation. This is a dual point of view, which we will make clearer in the next section by comparing the two generalizations.

Then we extend the notion of switch: for the switching of a generalized link, choose a partition in the first set, then choose a class in this partition, that is elect the upper edges belonging to this class, and adjacent to the corresponding vertex in the graph-form of the structure and erase all the other upper edges, then connect together, for each remaining class, its vertices (without making cycles!). The correctness criterion remains the same (i.e., we take Theorem 4 now as a definition of correction).

Definition 7. A proof-structure is correct iff for any choice of the (generalized) switches, the associated graph is acyclic and connected.

As with the sequential connectives we have to state a condition which ensures that cut-elimination is always defined. A step of normalization is the obvious generalization of the binary case (see Subsect. 1.2): substitute to a cut between $C(A_1, \ldots, A_n)$ and $C^*(A_1^\perp, \ldots, A_n^\perp)$, n cuts between $A_1, A_1^\perp, \ldots, A_n, A_n^\perp$. We first give the definition and then prove that it is correct with respect to normalization.

Definition 8. A n-ary multiplicative connective in proof-nets consists of two sets of partitions P and Q over $\{1, \ldots, n\}$, such that $P \perp Q$ and $P^\perp \perp Q^\perp$.

Lemma 2. *A proof-structure obtained after one step of reduction of a correct one is still correct.*

Proof. Take a proof-net with a cut between C and C^*, and consider the two following modules: the first M_1 is defined by the three vertices corresponding to the C-link, the C^*-link and the cut, while the second M_2 corresponds to the remaining of the structure. A partition r in the type of M_1 is obtained in this way:

First choose $p \in C$ and $q \in C^*$, say $p = \cup C_i$ and $q = \cup C_j^*$. Then

$$r = \bigcup_{i \neq i_0} C_i \bigcup_{j \neq j_0} C_j^* \cup (C_{i_0} \cup C_{j_0}^*),$$

for any choice of i_0 and j_0.

This entails that any s in M_2 is of the form $s_1 s_2$ where s_1 is only defined on the p-border and s_2 over the q-border (with obvious notations). Otherwise the starting proof-net would be incorrect. Furthermore $s_1 \perp p$ and $s_2 \perp q$. Hence by the Def. 8 above $s_1 \perp s_2$, and the proof-structure obtained after one step of reduction is correct by Theorem 5.

3.5. Decomposable Connectives

As in the sequential case there are decomposable connectives.

Definition 9. A connective is decomposable iff its first set P is the type of a formula-tree only built with binary connectives, and its second set Q is the type of the dual tree.

Proposition 2. *A decomposable connective satisfies Def. 8, i.e., is a particular case of general multiplicative.*

Proof. This proposition just states the consistency of the Def. 9 with the general Def. 8. We have first to check that $P \perp Q$. Take for any decomposable connective X the proof of the non-atomic axiom $X \wp X^\perp$. This is a proof-net where the X-tree is connected to its dual via axiom links. Retract in any correction subgraph of this proof-net all the inner edges of each formula-tree, and erase the axiom vertices. You clearly find a $G(p, q)$ graph for some partitions of P and Q. Hence $P \perp Q$.

Second we have to check that $P^\perp \perp Q^\perp$.

Lemma 3. *For any partition p, there is a decomposable connective $X(p)$ whose first set is the only partition p.*

Just make *times* between points in the same class and then *par* between these classes.

Now we prove the second point: choose a s_1 in $P^\perp$, and consider the proof-net obtained by linking the X and $X(s_1)$-trees. Surely it is correct. Do the same for s_2 in $Q^\perp$, and cut the two proof-nets by their dual formulas X and $X^\perp$. The usual normalization (binary case) yields a correct proof-net, therefore $s_1 \perp s_2$. Hence $P^\perp \perp Q^\perp$. That's what we wanted.

The same remarks, we made in the sequential case, apply, except that there is no more packing problem in this syntax. Hence, we cannot expect every correct proof-net to be sequentializable. (For example the non-atomic axiom with conclusions X and $X^\perp$ is correct, while we noted that it is not provable in the sequential case, unless one takes it as an axiom ...). Let us now introduce a notion for discussing a final point in this section.

Definition 10. A connective defined by P and Q is reflexive when $P^{\perp}=Q$, and $Q^{\perp}=P$.

This definition makes sense in both syntaxes, but is a priori not the same. Shall we ask that our multiplicatives in proof-nets be reflexive? The answer is a twofolded no.

First we know an example of a decomposable connective which is not reflexive:

$$X(1,2,3,4,5,6)=((1\otimes 2)\wp(3\otimes 4))\wp(5\wp 6)$$

has type:

$$P_X=\{\{\{1,2\},\{3,4\},\{5\},\{6\}\}\},$$

while the partition

$$\{\{1,3\},\{2,5\},\{4,6\}\}\in P_X^{\perp}$$

is not in Q_X, for all its elements have 5 and 6 in the same class. (This shows by the way that reflexivity is a stronger condition than the one of Def. 8).

Second, two modules of types P and Q such that $P^{\perp}=Q^{\perp}$ behave the same way, i.e., they are connectable to the same modules. Hence any time we have a non-reflexive connective its completion does not add any information. We should be glad to have a shorter description of its potential compositions, and not try to complete it.

4. Comparing the Two Kinds of Multiplicatives

The first thing of note is that we used the same objects, namely partitions, for both syntaxes, as well as the same notion of orthogonality. But there is even a stronger link. For instance:

$$P_{\wp}=\{\{1\},\{2\}\}=\{\{1,2\}\}^{\perp}=O_{\otimes}$$

and the correct statement in general is the following:

Proposition 3. *Let X be a decomposable connective, and P_X be its type in proof-nets while O_{X*} is its set of left rules, then $P_X\subseteq O_{X*}$.*

Proof. We prove it by induction on the structure of X. Hence there are two cases. If $X=X_1\wp X_2$, then any p in P_X is of the form p_1p_2, with obvious notations. Hence we can, starting with p as organization of the premises, derive X^* in the following manner: derive X_1^* starting with p_1 and X_2^* with p_2 by the induction hypothesis, and then make a *times* rule. If $X=X_1\otimes X_2$, then any p in P_X is of the form $p_1p_2(i,j)$ for some i and some j, where the notation means that we concatenate the two partitions and mix the two classes containing i and j. Then derive X^* by first deriving say X_1^* starting with p_1 where the i-class, one of the premise, contains as a context the atoms of the class containing j. Now derive X_2^* with p_2 where the j-class contains as a context X_1^*, and eventually make a *par* rule. The first step of induction was given above as an example.

Proposition 4. *Let X be a decomposable connective, and P_X be its type in proof-nets while O_X is its set of right rules, then $O_X = P_X^\perp$.*

Proof. Given p in O_X, we built the following sequential proof: the last rules will introduce $X(A_1, \ldots, A_n)$ with a binary combination that starts with p as organization. This is possible precisely because X is decomposable. Then we prove each premise by the following:

$$\frac{\vdash A_1, A_q^\perp \ldots \quad \ldots \vdash A_{i_q}, A_{i_q}^\perp}{\vdash A_q, \ldots, A_{i_q}, \otimes(A_q^\perp, \ldots, A_{i_q}^\perp)}$$

where we used a generalized times-rule. This is obviously a correct proof. We translate it by the usual way. Hence we now have a proof-net containing two parts, the X-tree connected by n axiom links to the remaining part, which is a module of type $\{p\}$. Hence $O_X \subseteq P_X^\perp$. Conversely, let p be in $P_X^\perp$. We take the X-tree and complete it to get the same proof-net as above, but this time grouping of dual atoms is made along p. This is a proof-net by definition. Hence we can sequentialize it. Last we remark that all the *times* rules involving the dual atoms can be made first, and only then the rules constructing X. This is a refinement of the sequentialization Theorem 2 which is a consequence of the following easy lemma which we shall not prove. Therefore p is in O_X.

Lemma 4. *If we allow arbitrary axioms in the proof-structures, then correct ones are still sequentializable.*

By arbitrary axioms we mean axiom links between any number of atoms. The Proposition 4 we just proved entails:

Corollary 1. *All decomposable connectives are reflexive in the sequential syntax.*

And this together with the example of a non-reflexive decomposable connective in the case of the proof-nets, which we gave in last section entails that inclusion in the Proposition 3 cannot be extended to an equality in general. As we noted previously this should be interpreted to the advantage of proof-nets, as this means that types are sometimes more economic descriptions for the same benefit. We completed our tour. We defined almost the same connectives in both syntaxes, yet they have more satisfying properties in the proof-nets.

5. Comparing with Existing Theory

5.1. Proving that the Two Correctness Conditions are the Same

We pretended in Theorem 4 that our correctness condition was equivalent to the usual one. Let us prove it. We start with a proof-structure in graph-form. First we show how to associate to each correction subgraph a family of trips. Duplicate each edge of the subgraph and give them opposite orientations. Then choose for each *times* vertex a orientation, left or right. Make a trip according to the following instructions:

– if you encounter a *par* vertex then keep on moving chosing the edge with same orientation as the one you left,

– if you encounter a *times* do the same except that you chose the edge to the left if this vertex is left-oriented, and else the other one.

Moves are obvious in the case of axioms conclusions. This procedure yields a family of trips which are only differing by the way we restored *times* switches. It's sufficient now to prove that the subgraph is acyclic and connected iff all associated trips are long ones. The if part is obvious, just glue any trip on itself for each visited formula. We prove the converse by induction on the number of vertices, that is on the number of links in the proof (including the special conclusion links). If the graph is acyclic and connected, we can find in it a vertex with only one incident edge, which is a conclusion. There are two main cases. If this vertex is connected to a *par* one, retract the graph along the linking edge. The trips in the retracted graph are long ones by induction hypothesis, and any trip in the original graph is obtained from those by merely adding the visit to the conclusion, and is evidently still long. If this vertex is connected to a *times* one, say v then three edges spread out of v. One for the conclusion we distinguished, and one for each premise of the *times* rule. Erase all the three and forget v as well as the conclusion. This yields a graph which has two connected components, one for each premise, because we supposed that the original graph was tree-like. Remark now that a trip in the whole structure is the combination of two trips in the connected components plus an orientation of v, and the visit to the conclusion. This combination clearly preserves longness.

This shows why one can spare the inspection of *times* switches, and hence shorten the checking of a proof-structure. Moreover the proof stresses the difference between two kinds of short trips. Those which corresponds to cyclic correction subgraphs, and which are definitely bad ones, and those which corresponds to only disconnected graphs expressing some kind of absence of communication between parts of the proof and which deserves a better treatment (see [Da 88]). But this also proves an interesting property of trips which we now study in detail.

5.2. Bilaterity

A trip can be represented as a sequence of the formulas it visits. As a trip is by its very nature cyclic, the starting point of this sequence is of no interest.

Definition 11. We say that a long trip is bilateral iff for any formulas A and B visited you never find this: $\ldots A \ldots B \ldots A \ldots B \ldots$.

We could have stated this in a other way. Write the sequence of visited formulas on a circle, and link each couple A^+, A^- by a diameter. The trip is bilateral iff no two diameters cross each other.

Corollary 2. *Long trips in a proof-structure are bilateral.*

This is a consequence of the preceding proof. This an important property, mainly because of its applications inside the theory, and we shall give some of them. But we can already comment it on an intuitive basis. For this we must have an idea of the link between trips and execution. We think of the trips as static and sequential checks of an execution of the structure, which is parallel. In other words, trips are sequential readings of a graph, while evaluation should proceed through it in a parallel way. Then we could say that bilaterity expresses a restriction to the parallelism one can expect from proof-nets, namely that proof-nets are descriptions of communications which one can sequentially simulate. This operational sequentiality would echo the static one of Theorem 2.

The synchronization lemma for *times* is proved in [Gir 87], and is extremely useful for the proof of sequentialization. It's an easy consequence of bilaterity.

Lemma 5. *The visiting of a times link by a long trip looks like:*

$$\dots A^- A\otimes B^- \dots A\otimes B^+ B^+ \dots B^- A^+ \dots .$$

This property proved also useful for the generalization of proof-nets to the case of quantifiers [Gir 88b]. Therefore we don't want to lose it in a attempt to generalize multiplicatives. We show in the following that if our generalization satisfy it, the first one suggested by Girard in [Gir 87b] fails to keep this property.

5.3. The Two Notions of Orthogonality

The generalization proposed in [Gir 87b] relied on the representation of connectives by sets of permutations. For the same purpose of solving the modularity problem he was lead to introduce the first notion of orthogonality.

Definition 12. Two permutations are orthogonal iff their composition is circular.

Note that the way you compose the permutations is in different, for if $\sigma\tau$ is circular, then it's true for $\tau\sigma$. This definition is the analog of Def. 2. Yet they are not the same. To state this precisely, remark that a permutation can always be impoverished in a partition, by simply forgetting the inside structure of its cycles. Hence we can say that two permutations are orthogonal in the sense of Def. 2, in which case we shall say, only for the purpose of comparing the two notions, that they are *strongly* orthogonal.

Proposition 5. *Two strongly orthogonal permutations are orthogonal.*

Proof. By induction on the cardinal of their domain. If they are strongly orthogonal, then one of them, say σ has a cycle with only one point, say x, which is in the graph $G(\sigma, \tau)$ a terminal edge. Hence the two restricted permutations $\sigma - \{x\}$ and $\tau - \{x\}$ are still strongly orthogonal, and orthogonal by the induction hypothesis. This entails that σ and τ themselves are orthogonal, because σ behaves like the identity on x. The proof underlines a difference between the two notions, namely that the strong orthogonality is easily defined by induction, whereas it seems much more problematic for the other.

So, strong orthogonality is a stronger condition! But the following proposition shows precisely what is required in addition for two orthogonal permutations to be strongly orthogonal. For two orthogonal permutations σ and τ on the same set $\{1, \dots, n\}$ we define their *communication sequence* to be:

$$(1, \sigma(1), \tau\sigma(1), \sigma\tau\sigma(1), \dots, (\tau\sigma)^{n-1}(1), \sigma(\tau\sigma)^{n-1}(1)).$$

We can interpret this sequence as the order in which a trip encounters the common border formulas of two modules switched respectively by σ and τ. Hence it makes sense to say that such a sequence is bilateral.

Proposition 6. *Two orthogonal permutations are strongly orthogonal iff their communication sequence is bilateral.*

Proof. We again prove this by induction on the cardinal of their domain. For the if part, take an x in the sequence which is followed by itself. This surely exists. Then erase both of its occurrences. This yields the communication sequence of the two restricted permutations $\sigma - \{x\}$ and $\tau - \{x\}$. By induction hypothesis these two are strongly orthogonal, hence this still holds for the original ones. For the converse part, erase a terminal edge, say x in $G(\sigma, \tau)$. This yields two restricted permutations which have a bilateral communication sequence. This also holds for σ and τ for their sequence is only an insertion of two consecutive x in the other.

5.4. Conclusion of the Comparison

Permutations carry an idea of dynamical behavior which allowed Girard to formulate a normalization algorithm based on iteration of permutations (see [Gir 87b]). Our representation of multiplicatives does not try to compete with Girard's one with respect to normalization, i.e., it doesn't make any progress (nor does it regress!) with respect to dynamics in proof-nets. We only claim it to be an improvement of the static aspect of the problem. First because partitions are less cumbersome objects than permutations in general, so we can pretend to have simplified the descriptions of multiplicatives at work. The plugging condition itself given in Theorem 5 get simplified because our orthogonality is simpler (it's inductively definable). Second, because we enclosed in our definitions only the meaningful multiplicatives, namely those with bilateral communication sequences. This is the meaning of Proposition 6.

Let us end this comparison with an example of connective which we reject. We note σ^+ the following permutation: $\sigma^+(i) \equiv i+1[\text{modulo } 3]$. Then the ternary connective defined by the sets $\{\sigma^+\}$ and $\{\sigma^+, i\}$ is a correct one in Girard's sense. Because σ^+ belongs to its orthogonal. Indeed it's perfectly sound to extend normalization to such a connective. Yet the communication sequence associated to a plugging of σ^+ on itself is $(1, 2, 3, 1, 2, 3)$, which is highly non bilateral.

Conclusion

We saw with some surprise that the realm of multiplicatives became quite complex, even handled by a careful generalization. Yet the generalization seems more natural in the non-sequential framework (because of the packing problem in the sequential case). Maybe we witness here the limits of sequential presentations of logic.

The philosophy of this study was, to confine generalization where it is suggested by the current theory and intuitively supported. For example, modules are already generalized connectives. Hence we do not fear to say a word of a further generalization which is reasonnable in this sense. It is the one connected with the mix rule which we showed in Subsect. 3.3, discussing about modules with void borders. It also arises quite naturally if one weakens the notion of orthogonality in Def. 2. This seems to be a worthwhile generalization, especially if the work in progress regarding normalization in proof-nets shows, as it is plausible, that mix has a computational meaning.

Another question of interest is the possibility of an extension of *coherence semantics* (see [Gir 87]) to these new connectives. Van de Wiele gave recently a solution to this question, formulated with the help of game theory (see [VdW 88]).

Acknowledgements. The authors wish to thank A. Fleury who gave them Lemma 1, J. Van de Wiele who reported a mistake in a preliminary version, and J. Y. Girard who encouraged them many times.

References

[Gentzen 69]	Szabo, M.E.: Collected works of Gehrard Gentzen. Amsterdam: North-Holland 1969
[Gir 87]	Girard, J.Y.: Linear logic. Theoretical Computer Science **50** (1987)
[Gir 87b]	Girard, J.Y.: Multiplicatives. To appear in the Proceedings of the Congress of Logic and Computer Science held in Torino, October 1986
[Gir 88]	Girard, J.Y.: Towards a geometry of interaction. To appear in the acts of AMS conference on categories, Boulder, June 1987
[Gir 88b]	Girard, J.Y.: Quantifiers in linear logic. To appear in the Proceedings of the SILFS conference, held in Cesena, January 1987
[Da 88]	Danos, V.: Correctness of proof-nets. Draft, université de Paris VII, March 1988
[VdW 88]	Van de Wiele, J.: Jeux en Logique Linéaire. Draft, Université de Paris VII, November 1988

Received May 31, 1988/in revised form November 22, 1988