Electronic Notes in Theoretical Computer Science 17 (1998)
URL: http://www.elsevier.nl/locate/entcs/volume17.html 20 pages

Proof nets Construction and Automated Deduction in Non-Commutative Linear Logic - extended abstract -

D. Galmiche

LORIA UMR 7503 - Univ. Henri Poincaré
Campus Scientifique, BP 239,
54506 Vandœuvre-lès-Nancy Cedex, France

B. Martin

Ecole Normale Supérieure de Lyon
46, Allée d'Italie, 69364 Lyon Cedex 07, France

Abstract

Proof nets can be seen as a multiple conclusion natural deduction system for Linear Logic (LL) and form a good formalism to analyze some computation mechanisms, for instance in type-theoretic interpretations. This paper presents an algorithm for automated proof nets construction in the non-commutative multiplicative linear logic that is useful for applications including planning, concurrency or sequentiality. The properties of this algorithm can be proved from a recently defined graph-theoretic characterization of non-commutative proof nets. Involving simple construction principles improved in the commutative case, it leads also to a new proof search method for the non-commutative fragment. Moreover because of the relationships between the non-commutative linear logic and the Lambek calculus we can derive from it an alternate method for automatic construction of proof nets in this calculus.

1 Introduction

Linear Logic (LL) can offer a framework to study and analyze various notions or approaches of programming and computation [11]. Many developments and applications based on the *proofs-as-programs* or *proof-search as computation* paradigms require adequate contributions on proof search in given LL fragments, and consequently new methods for proof or proof nets construction that could be used in logical frameworks or type-theoretic languages. Automated proof search can be investigated with definitions of classes of proofs that are complete w.r.t. provability (as uniform, focusing or canonical proofs [10]) or alternatively with the notion of *proof net* that is a concrete structure introduced to cope with the problems arising from the intrinsic parallelism of the linear sequent calculus, i.e., a counterpart of natural deduction in LL. We have defined an algorithm for automated proof net construction for the Multiplicative Linear Logic (MLL) and shown that it could naturally provide a connection (proof search) method for this logical fragment [7]. In fact, automated theorem proving or verification can strongly benefit from such investigations on proof nets.

In this paper, we consider the multiplicative fragment of Non Commutative Linear logic (NCMLL) [1,2] and we design an algorithm for automatic construction of non-commutative proof nets that could be also viewed as a proof search method in this logical fragment. Such results are important for theorem proving in applications including planning, concurrency, sequentiality or computational linguistics and also for verification of concurrent distributed systems. In fact, the order sensitiveness of the NCMLL calculus is reflected by planarity conditions on proof nets, justified by a graph-theoretic characterization of non-commutativity [18]. A main point is to keep the construction principles used in the MLL case that lead to a direct construction of proof nets without an *a posteriori* verification [7].

Moreover because of the relationships between Lambek calculus and non-commutative linear logic [3], another interesting result is that we can derive, from the previous results, an alternative algorithm for construction of Lambek proof nets [17,20] that is not based on the explicit use of labels on the links or on some unification on such labels.

2 Non-commutative Linear Logic and Proofs

Linear logic introduced by J.-Y. Girard [11] keeps one of the structural rules of the sequent calculus, namely the *exchange rule* that allows to disregard the order of the assumptions and conclusions of a proof. The usual linear logic including this rule may be called *commutative linear logic* because it allows to prove the commutativity of the multiplicative connectives. The aim to see what happens when we remove this rule from the linear sequent calculus leads to the study of the logic called *non-commutative linear logic*. The sequent calculus and phase semantics in the propositional case have been investigated in [1] and other studies on non-commutative classical linear logic were developed with a restricted form of exchange rule (cyclic exchange rule or schift rule) [12,21].

The NMLL system [1].

We present the multiplicative fragment of non-commutative linear logic, called here NMLL, knowing that we can easily and naturally extend it to the additive connectives to obtain NMALL [2].

Definition 2.1 (formulae)

Formulae of MNLL are defined as follows:
(i) if P is a propositional letter and $n \in Z$, (P^n) is a formula of MNLL.
(ii) if A and B are formulae of MNLL then $A \otimes B$ and $A \wp B$ are formulae of MNLL.
if P is a propositional letter and $n \in N$ then P^0 is P, P^{+n} is $P^{\perp \ldots \perp (n\ times)}$ for $n > 1$ and P^{-n} is ${}^{\perp \ldots \perp (n\ times)}P$ for $n > 1$.

A *sequent* of MNLL is any configuration $\vdash \Gamma$ where Γ is a finite sequence of formulae of MNLL.

Definition 2.2 (negation of formulae)
For each formula A of MNLL, we define $A^\perp$ (*linear post-negation*) and ${}^\perp A$ (*linear retro-negation*) as follows:
(i) if P is a propositional letter and $n \in Z$, $(P^n)^\perp = P^{n+1}$ and ${}^\perp(P^n) = P^{n-1}$
(ii) $(A \otimes B)^\perp = B^\perp \wp A^\perp$, ${}^\perp(A \otimes B) = {}^\perp B \wp {}^\perp A$
(iii) $(A \wp B)^\perp = B^\perp \otimes A^\perp$, ${}^\perp(A \wp B) = {}^\perp B \otimes {}^\perp A$
If A is a formula of MNLL and $n \in Z$ then A^n will denote A if $n = 0$, $A^{\perp \ldots \perp (n\ times)}$ if $n > 0$ and ${}^{\perp \ldots \perp (n\ times)}A$ if $n < 0$.

We can verify that for each formula A of MNLL $({}^\perp A)^\perp = {}^\perp(A^\perp) = A$

Definition 2.3 (sequent calculus)
We define the one-sided sequent calculus for MNLL as follows:

$$\frac{}{\vdash A^\perp}\ Id \qquad \frac{\vdash \Gamma, A, \Gamma' \quad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta, \Gamma'}\ Cut_1 \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, A^\perp, \Delta'}{\vdash \Delta, \Gamma, \Delta'}\ Cut_2$$

$$\frac{\vdash \Gamma, A, \Gamma' \quad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta, \Gamma'}\ \otimes_1 \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, B, \Delta'}{\vdash \Delta, \Gamma, A \otimes B, \Delta'}\ \otimes_2 \qquad \frac{\vdash \Gamma, A, B, \Gamma'}{\vdash \Gamma, A \wp B, \Gamma'}\ \wp$$

If we restrict the rules of this sequent calculus to formulae of MLL and if we moreover consider $A^\perp$ in these rules as the metalinguistic linear negation of A in MLL and if finally we add the *exchange rule*, we get an alternative formulation of the sequent calculus for MLL equivalent to the one of [11]. More details on this MNLL system can be found in [2].

The MNCLL system [18].

If we add ${}^\perp A = A^\perp$ the relations $({}^\perp A)^\perp = {}^\perp(A^\perp) = A$ become $(A^\perp)^\perp = A$, having only one linear negation. We obtain with this addition the MNCLL system (with the same axioms and rules as MNLL) proposed in [18]. In this system the sequents $\vdash A^\perp, A$ and $\vdash A, A^\perp$ are both admitted as axioms but let us note that the latter is not an axiom (and even not a theorem) in MNLL.

Let us illustrate this system with the following sequent
$\vdash C \otimes (D \otimes A), A^\perp \wp B^\perp, B \otimes (D^\perp \wp C^\perp)$ and one of its proof in MNCLL.

$$
\dfrac{\dfrac{}{\vdash B, B^{\perp}}\,id \qquad \dfrac{\dfrac{\dfrac{}{\vdash C, C^{\perp}}\,id \quad \dfrac{\dfrac{}{\vdash D, D^{\perp}}\,id \quad \dfrac{}{\vdash A, A^{\perp}}\,id}{\vdash D \otimes A, A^{\perp}, D^{\perp}}\,\otimes_1}{\vdash C \otimes (D \otimes A), A^{\perp}, D^{\perp}, C^{\perp}}\,\otimes_1}{\vdash C \otimes (D \otimes A), A^{\perp}, D^{\perp} \wp\, C^{\perp}}\,\wp}{\dfrac{\vdash C \otimes (D \otimes A), A^{\perp}, B^{\perp}, B \otimes (D^{\perp} \wp\, C^{\perp})}{\vdash C \otimes (D \otimes A), A^{\perp} \wp B^{\perp}, B \otimes (D^{\perp} \wp\, C^{\perp})}\,\wp}\,\otimes_2
$$

If we consider the similar sequent where $D \otimes A$ is replaced by $A \otimes D$ then it is not provable in MNCLL. Compared to the previous proof the expected one would have the same schema but the last (from the bottom) $\otimes_1$ rule cannot be applied.
The previous system MNCLL is equivalent to the multiplicative fragment of Cyclic Linear Logic (CyMLL) [21] that includes a cyclic exchange rule and we can also recall that there exists a linear length translation from the multiplicative fragment of Abrusci's non-commutative linear logic into the multiplicative fragment of Yetter's cyclic linear logic [19]. In the rest of the paper we consider the MNCLL system to deal with non-commutative linear logic and use the notation NCMLL to denote this logic.

3 Non-commutative proof nets

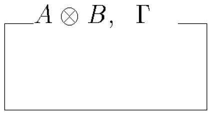
The notion of *proof net*, a new kind of syntax for linear logic, has been introduced by J. Y. Girard [11,13] as a counterpart of natural deduction in linear logic, in order to cope with the problems arising from the intrinsic parallelism of linear sequent calculus. This notion, originally restricted to MLL, is defined for different fragments as for instance MALL (extension with the additives) [14,15] or NCMLL [2].
A main point consists in finding the most adequate definition of proof net, that can be inductive or not. Before to define the notion of non-commutative proof net, we start to recall what is a (commutative) proof net of MLL [1] and some basic structures useful for the construction. To consider the construction of proof nets, we use simple concepts as the decomposition tree (that is trivially defined as the tree of all the subformulae of the given sequent), the leaves of which being free or not (w.r.t. axiom-links) during the construction process.

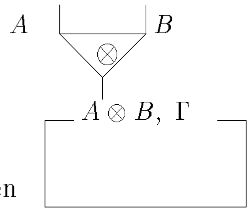
Definition 3.1 A *literal* is an atomic formula A or its negation $A^{\perp}$.
Two literals are *dual* if one is the negation of the other.
A *pre-proof structure* of a sequent $\vdash A_1, ..., A_n$ is defined in the following way:

(i) A sequence $A_1, ..., A_n$ is a pre-proof structure with $A_1, ..., A_n$ as leaves.

(ii) $\otimes$-link:

[1] In fact, we only consider MLL without constants, i.e., the MLL^{-} fragment

If 

is a pre-proof structure, then 

is a pre-proof structure;

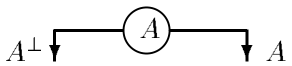
(iii) $\wp$-link : replace $\otimes$ by $\wp$ in the previous case.

The decomposition tree of a sequent can be inductively constructed but the essential for the proof net construction consists in finding the good links between dual leaves. In fact, a *proof structure* is composed by the decomposition tree with a set of axioms-links covering the leaves and there are various criteria to verify if a given proof structure is also a *proof net* [4,6,11]. A first approach could consist in building a proof structure and in verifying a posteriori if it is a proof net. Here we want to consider the proof net construction from an automated deduction point of view and then, from a given sequent, to directly and automatically construct a corresponding proof net, if there exists. For that, the use and the analysis of an inductive definition of proof nets, as for instance the one of [8], is helpful.
It is well-known that the proof nets of MLL are characterized by a simple and elegant graph-theoretic condition, saying that any Danos-Regnier graph (D-R graph) is a proof net of MLL if and only if it is acyclic and connected under a choice of $\wp$-link switching [2] (switching condition) (see [6] for more details). But what about the non-commutative proof nets, firstly defined in [2] ? We now present the inductive definition of non-commutative proof nets and the graph-theoretic characterization from which we can justify the properties of our algorithm.

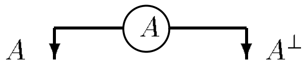
Inductive non-commutative proof nets.

We define a *non-commutative proof net* as a plane graph induced from proofs of MNCLL.

Definition 3.2 A non-commutative proof net (NCPN) is defined by induction as follows:
(1) *Axiom-1.* We draw an axiom-link, corresponding to $\vdash A^\perp, A$ as follows and so we obtain an elementary NCPN with $A^\perp, A$ as conclusions.

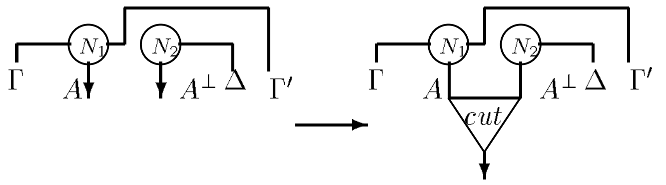


(2) *Axiom-2.* We draw an axiom-link, corresponding to $\vdash A, A^\perp$ as follows and so we obtain an elementary NCPN with $A, A^\perp$ as conclusions.

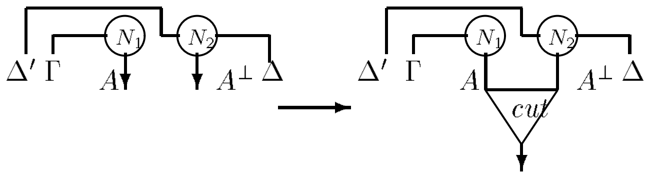


[2] for each $\wp$-link one selects one premise and removes its connection with the link.

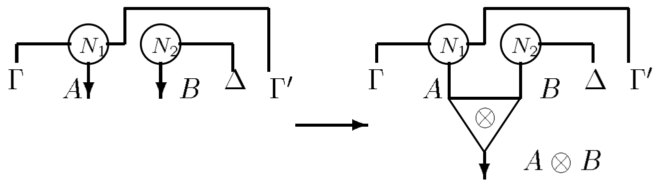
(3) Cut_1. If N_1 and N_2 are disjoint NCPN's with the multisets Γ, A, Γ' and $A^\perp, \Delta$ respectively as conclusions, we draw a cut-link as follows and so we obtain a NCPN with the multiset Γ, Δ, Γ' as conclusions.



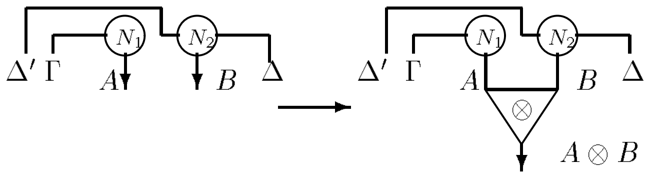
(4) Cut_2. If N_1 and N_2 are disjoint NCPN's with the multisets Γ, A and $\Delta', A^\perp, \Delta$ respectively as conclusions, we draw a cut-link as follows and so we obtain a NCPN with the multiset Δ', Γ, Δ as conclusions.



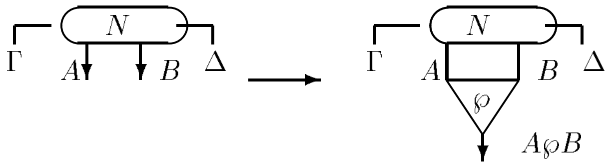
(5) $\otimes_1$. If N_1 and N_2 are disjoint NCPN's with the multisets Γ, A, Γ' and B, Δ respectively as conclusions, we draw a times-link as follows and so we obtain a NCPN with the multiset $\Gamma, A \otimes B, \Delta, \Gamma'$ as conclusions.



(6) $\otimes_2$. If N_1 and N_2 are disjoint NCPN's with the multisets Γ, A and Δ', B, Δ respectively as conclusions, we draw a times-link as follows and so we obtain a NCPN with the multiset $\Delta', \Gamma, A \otimes B, \Delta$ as conclusions.



(7) $\wp$. If N is a NCPN with the multiset Γ, A, B, Δ as conclusions, we draw a par link as follows and so we obtain a NCPN with the multiset $\Gamma, A \wp B, \Delta$ as conclusions.



A *plane proof net* of MLL is a commutative proof net without crossing in the graph drawing. From the previous definition, we know that any non-commutative proof net with Γ as conclusions is a plane proof net of MLL with Γ as conclusions. Moreover any proof net of a non-commutative version of MLL (MLL without the exchange rule) can be drawn as a plane graph.

A Graph-theoretic characterization [18].

Let us recall the graph-theoretic characterization of non-commutative proof net given in [18] with the main definitions and theorems.

Definition 3.3 ([6])
A *directed Danos-Regnier graph* (D-R graph) is a directed graph which consists of axioms-links (with two out-edges), cut-links (with two in-edges), times-links and par-links(with two in-edges and one out-edge) and conclusions nodes.

Theorem 3.4 *([6])*
A D-R graph is a proof net of MLL if and only if it is always acyclic and connected under any choice of par-switching (this global condition is called switching condition).

Let us recall now that a non-commutative proof net is a specific proof net of MLL and thus can drawn as a D-R graph.

Definition 3.5 ([18])
i) A *marked* D-R graph is a D-R graph where each $\otimes$-link and $\wp$-link has two in edges labeled L (left) and R (right) and one out-edge labeled C (conclusion).
ii) A marked D-R graph is said to be *uniformly directed* if the L-edge, R-edge and C-edge for a link is uniformly drawn in a fixed cyclic order for all the tensor and par-links.
iii) A marked D-R graph is said to be *strongly planar* if it is plane and uniformly directed.

If the sequent to prove (and thus the marked *D-R graph* G) has several conclusions $A_1, A_2, ...A_n$, we connect them by $\wp$-links to obtain an equivalent sequent (and a closure of G) with only one conclusion $A_1 \wp A_2, ... \wp A_n$. Thus we will construct a strongly planar proof net for this formula and then suppress these fictitious $\wp$-links from the proof net. Moreover, the previous definition can be naturally extended to the proof-structures, the condition on the cyclic order being only applicable to links with both premises treated.
In order to define a graph-theoretic characterization, [18] defines the two notions of *long trip condition* and of *stack condition*. The former was originally introduced by Abrusci in order to characterize his multiplicative non-commutative Linear Logic MNLL [1]. The main result is the following

Theorem 3.6 *[18] Let $\mathcal{DR}$ a marked D-R graph, the four following sentences are equivalent:*
1) $\mathcal{DR}$ represents a non-commutative proof net.
2) $\mathcal{DR}$ satisfies the switching condition *and is* strongly planar.
3) $\mathcal{DR}$ satisfies the switching condition *and the* long trip condition.
4) $\mathcal{DR}$ satisfies the switching condition *and the* stack condition.

Here we will use the equivalence between 1) and 2) to justify our proof net search method in NCMLL. From now a non-commutative proof net corresponds to the previous inductive definition. In fact, the Abrusci's definition

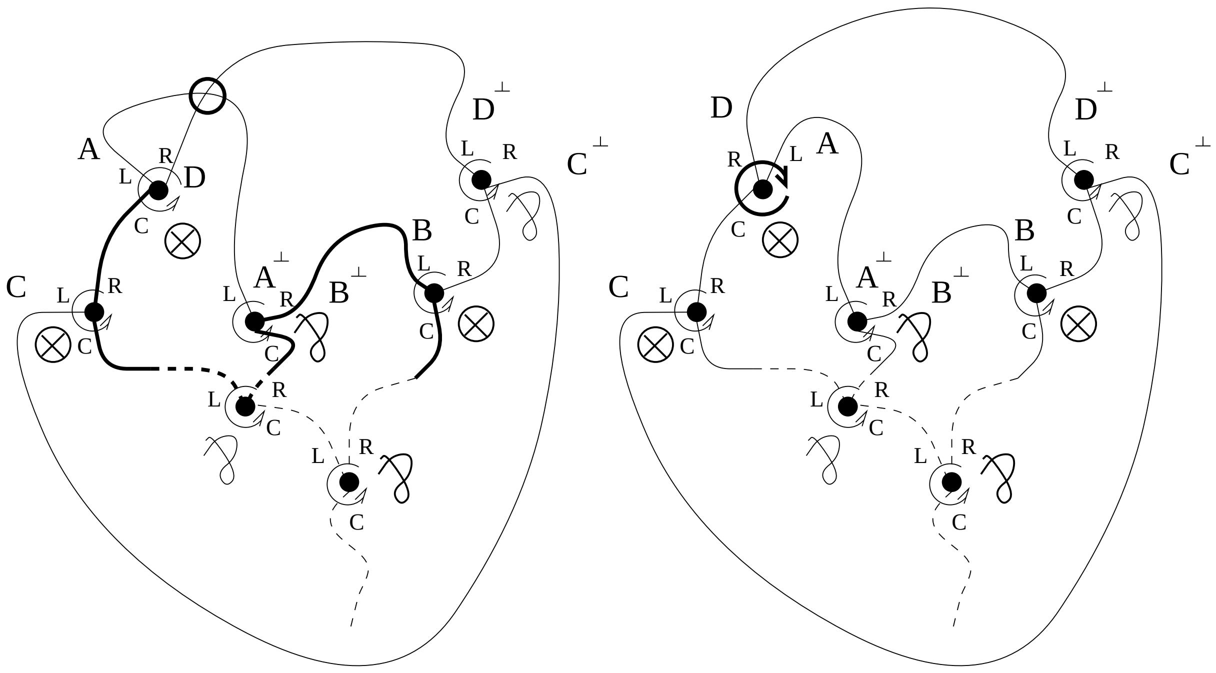


Fig. 1. Marked D-R graphs of $\vdash C \otimes (A \otimes D), A^{\perp} \wp B^{\perp}, B \otimes (D^{\perp} \wp\, C^{\perp})$

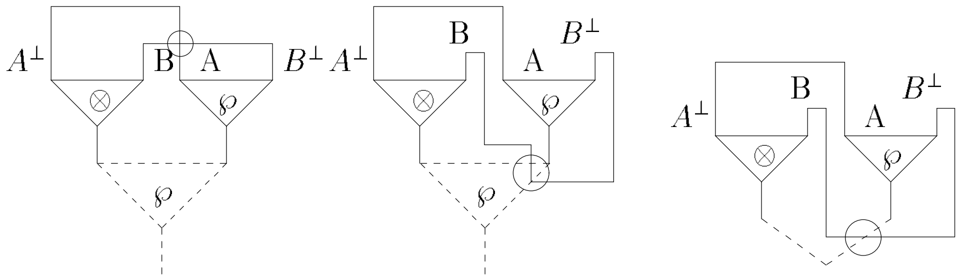
of non-commutative proof net corresponds to a marked D-R graph satisfying the switching condition and the long trip condition.
As illustration, the figure 1 presents two marked D-R graphs of the sequent $\vdash C \otimes (A \otimes D), A^{\perp} \wp B^{\perp}, B \otimes (D^{\perp} \wp\, C^{\perp})$ that are not strongly planar. The one on the lefthand side is not planar and the one on the righthand side is not uniformly directed (the link $A \otimes D$ is not in the fixed order). In fact, as already explained in section 2, this sequent is not provable in non-commutative linear logic.

4 Non-commutative proof net construction

Let us consider the following characterization: *a MLL proof net is a NCMLL proof net if and only if it is strongly planar (in the meaning of [18])*. It is in fact a corollary of the result saying that a marked D-R graph is a NCMLL proof net if and only if it verifies the *switching condition* and is strongly planar. In fact, we want to consider NCMLL proof nets as a particular classe of MLL proof nets and then to use or adapt, if possible, construction principles improved in the commutative case [7].

4.1 Preliminary remarks

We use from now the representation of proof nets that is based on the decomposition tree notion (see previous sections). It has the advantage to directly impose the condition on the fixed order (clock-wise or anti-clock-wise). Thus, our goal is to construct MLL proof nets the representation of which corresponds to a strongly planar graph. Let us consider the following sequent to prove $\vdash (A^\perp \otimes B) \wp (A \wp B^\perp)$, the corresponding proof net is not strongly planar and moreover it is not possible to find a planar representation from its decomposition tree.



We observe with this example that the planarity conditions, added by the fictitious $\wp$-links, consist in forbiding to cross the added edges linking all the conclusions to the fictitious root, i.e., intuitively in forbiding to pass under the conclusions. The added $\wp$-links will be not represented in the figures and the construction will be done directly from the multi-conclusion sequent.

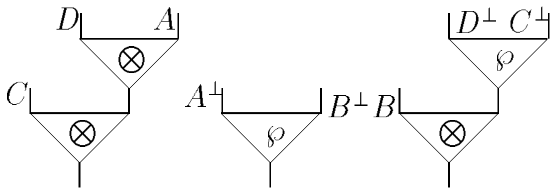
4.2 Construction principles

In this section, we will present some principles for the construction of proof nets that have been improved at first in [8] for MLL. The construction of proof nets can be done as the construction of proof structures with a validation through the application of various correction criteria. But here we want to directly construct proof nets without a posteriori verification.

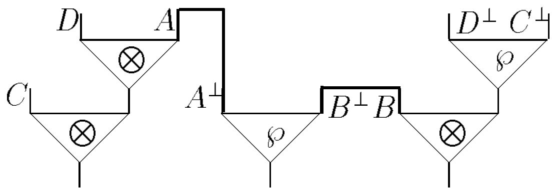
Let us recall that there exists a polynomial algorithm that returns that a proof structure is a proof net or not and that the decision problem for MLL is NP-complete [16].

The main point here consists not in building a complete proof structure and then in verifying if it is a proof net or not, but in verifying dynamically step by step that the structure under construction would finally be a proof net, if there exists one. In fact, we can detect that it could not be the case and then can stop the non-necessary research of other axiom-links. The first step consists in building the decomposition tree Θ and in constructing from Θ a set of sub-proof structures $\mathcal{R}$ (in fact sub-proof nets) that finally will contain the final proof net if there exists. At the initial step we have $\mathcal{R} = \emptyset$. At each step, the algorithm arbitrarily consider a free leaf A (resp. $A^\perp$) belonging to a branch of Θ where some leaves are already treated, with as main constraint to avoid the construction of too many distinct sub-proof nets in $\mathcal{R}$, by merging

as soon as possible some elements of $\mathcal{R}$. Then, for each dual free leaf $A^\perp$ (resp. A), we link them with an axiom-link, that is an elementary proof net that we add to $\mathcal{R}$. Then the `Propagation_construction` procedure, below explained, is applied to A (resp. $A^\perp$) and then, if there is no failure, to $A^\perp$ (resp. A). This procedure considers a subformula A from Θ , called the current formula, and while there exists a link L having this current formula among its premises and such that the other premise B of this formula is treated, then if L is a $\otimes$-link and its premises belong to distinct elements (nets) of $\mathcal{R}$, these elements are merged into a unique element having $A \otimes B$ among its conclusions that is added to $\mathcal{R}$, replacing the initial nets, else the procedure fails. If L is a $\wp$-link and its premises belong to the same element of $\mathcal{R}$, then the $\wp$-link is added to it and else added in a waiting list. This propagation process is essential here to cover the semantics of the connectives during the proof net construction. More details can be found in [7]. To summarize, the decomposition tree is a frame we used to construct the expected proof nets from elementary and smaller proof nets with the `Propagation_construction` procedure. It is justified by the semantics of the links, more deeply the meaning of *1-links* (like $\wp$) and *2-links* (like $\otimes$). In fact the underlying proof-theoretical ideas, already present in [8], consists in founding the construction of proof nets on the notion of subnets, following a dual approach of [5].
Let us consider the sequent $S\ :\vdash C \otimes (D \otimes A), A^\perp \wp B^\perp, B \otimes (D^\perp \wp\ C^\perp)$ and the corresponding proof net construction. The decomposition tree Θ of S is the following:



If the first free leaf chosen by the algorithm is A or $A^\perp$ and that the second one is $B^\perp$ (thus $\mathcal{R}$ is composed of the links $(A, A^\perp)$ and $(B, B^\perp)$), then the procedure `Propagation_construction` fails because the premises of the $\wp$-link do not belong to the same element of $\mathcal{R}$.



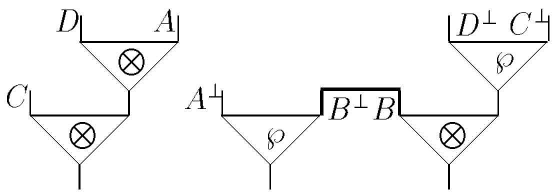
But, in fact, this sequent S is provable in MLL (proof in 2.2) and also in NMLL. Here and in the next examples of proof nets construction, the **links** and the **subnets** of $\mathcal{R}$ obtained by the `Propagation_construction` procedure are represented **in bold** in the figures. We can then identify the various steps of the algorithm execution.

As illustrated here it is necessary to determine, if possible, some orders in the treatment of formulae (and thus in the treatment of free leaves) that could surely lead to a proof net if the sequent is provable. The basic principle, induced by the induction definition and the semantics of the *1-links* and *2-links*, consists in treating, from the decomposition tree, as soon as possible the 2-links, to avoid, at each step, the exploration of unuseful branches and also the construction of too many isolated subnets in $\mathcal{R}$.
In fact, we have to postpone if necessary the treatment of the $\wp$-link (the 1-link), the premises of which belong to distinct subnets of $\mathcal{R}$ but that could be perhaps merged later. Regarding the permutability results in linear sequent calculus [9], it is always possible during a bottom-up proof search, when we have a choice between a $\otimes$ or a $\wp$ rule (or link), to start with the latter. Thus naturally, the proof nets being constructed with a dual top-down strategy, the corresponding dual result consists in starting, during this construction with the $\otimes$-links when it is possible.

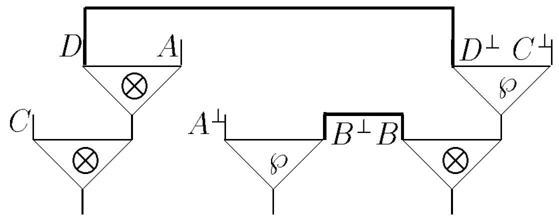
4.3 The non-commutativity.

Our goal is to keep the construction principles that work well for the commutative case and to take into account the particularities due to the non-commutativity as independently as possible. Regarding the graph-theoretic characterization of [18], the basic idea consists in allowing, during the construction process, the creation of some axiom-links only if the proof structure under construction remains strongly planar. With such "invariant condition" the final proof net will be strongly planar and thus will be a non-commutative proof net. Consequently, a simple method consists in defining a "successor" permutation on the set of the free leaves, the orbits of which are the set of free leaves that could be connected with an axiom-link, with preservation of the strong planity of the corresponding graph.

Let us illustrate this point with the following example:



In this net under construction the unique orbit of the permutation is (C D A $A^\perp$ $D^\perp$ $C^\perp$).
If we now consider the following net



then there are only two possible orbits, namely $(C\ C^{\perp})$ and $(A\ A^{\perp})$.

Consequently, we define the `Next_free_leaf` function that calls another function named `Next_atomic_formula` that returns the next atomic formula in a depth-first traversal of the extended (with fictitious edges) proof structure. The so-called "successor" of the last atomic formula being the first next one. This function can be implemented, for instance, with the cyclic list of the atomic formulae that can be naturally generated during the construction of the decomposition tree.

```
Function Next_free_leaf (A : free_leaf ) → free_leaf;
   A := Next_atomic_formula(A);
   While (A is not a free leaf) do :
      A := atomic formula linked to A by an axiom link ;
      A := Next_atomic_formula(A) ;
   EndofWhile
   Return (A) ;
```

Fig. 2. The `Next_free_leaf` function

Therefore, we iterate this function call (see Figure 2) on the free leaf we want to deal with and thus we exactly collect the possible candidates that would surely lead to a planar proof net.
An important remark is that it is in fact sufficient to consider the odd occurrences in this set of candidates. Otherwise, the function `Next_free_leaf` would generate a cycle with an odd length and the free leaves of this cycle could not be associated as pairs. Moreover, it is arbitrary to choose one given order for this decomposition tree traversal. But it appears natural to go to the left (resp. right) from the left (resp. right) premises of $\otimes$-links, taking into account the $\otimes$-rule.

4.4 Description of the algorithm

The following algorithm, with the principal procedure named NMLL_PN, deals with three main data-structures:
- a list $\wp$`_waiting` of $\wp$-links having both premises treated;
- a list $\wp$`_to_treat` of $\wp$-links, the premises of which are not treated (when they are introduced in the list);
- a list `f_to_treat` of subformulae in the structure under construction the branches of which are to be treated;
It is composed of the following four procedures. The initial call on the input

sequent S of the main procedure, the `NMLL_PN` one, will return either a failure message or a non-commutative proof net of S. Let us precise that the arguments of the procedures are called by value except the ones after "variables" that are called by address. In fact, we will consider here an extended version of the procedure of figure 2, namely `Next_free_leaf` (A: free_leaf, dir: boolean) $\rightarrow$ free_leaf; dir indicating the direction of the traversal. In this case, the lists `f_to_treat` et `℘_to_treat` include pairs with indication of the direction during the traversal.

Procedure `NMLL_PN` (S: NCMLL sequent) $\rightarrow$ `failure` or Π: NCMLL proof net

(i) construct the decomposition tree Θ of S;

(ii) initialize `f_to_treat`, `℘_to_treat` and `℘_waiting` to $\emptyset$;
one conclusion of S, arbitrarily chosen, is added to `f_to_treat`;

(iii) $\mathcal{R} := \emptyset$;

(iv) $\mathcal{R}$:= `Search_free_leaf_to_treat(`$\mathcal{R}$, `f_to_treat`, `℘_to_treat`, `℘_waiting`);

(v) Let Π be the unique element of $\mathcal{R}$, Return(Π).

Let us describe now the three main procedures used by this procedure.

Procedure `Search_free_leaf_to_treat(`$\mathcal{R}$, `f_to_treat`, `℘_to_treat`, `℘_waiting`) $\rightarrow$ `failure` or $\mathcal{R}$

(i) While one of the three lists is not empty, do
  (a) if `f_to_treat` is not empty, then consider its first element (A, dir) (that is removed from the list);
    - if A is a free leaf then `Treat_free_leaf(A,` $\mathcal{R}$, `f_to_treat`, `℘_to_treat`, `℘_waiting`);
    - if A has $\otimes$ as principal connective and has L and R respectively as its left and right premises then insert (L, dir_L) and (R, dir_R), at the beginning of `f_to_treat`;
    - if A has $\wp$ as principal connective then if its both premises were not treated, (A, dir) is added to `℘_to_treat`;
  (b) else if `℘_to_treat` is not empty then consider its first element; remove it from the list; if the both premises of this $\wp$-link are not already treated then add one of them in `f_to_treat` (with the same direction);
  (c) else the list `℘_waiting` being not empty, consider its elements doing $\mathcal{R}'$:= `Propagation_construction(`A, true, $\mathcal{R}$, `f_to_treat`, `℘_waiting`) if A is one of the premises of the $\wp$-link ; if all elements are considered without success, then return(`failure`) else return($\mathcal{R}'$)[3];

(ii) If all the leaves are treated[4], then return($\mathcal{R}$) else return(failure).

[3] Let us remark that $\mathcal{R}'$ can be a failure.
[4] We have necessarily $|\mathcal{R}| = 1$ and the structure under construction is a proof net

Procedure `Treat_free_leaf` (A : free leaf, variables $\mathcal{R}$, `f_to_treat`, $\wp$`_to_treat`, $\wp$`_waiting`)

(i) B := `Next_free_leaf` (A, dir);
$\mathcal{R}'$:= `failure`;

(ii) Repeat

(a) if B is a dual leaf of A then,
create an axiom-link between A and B;
$\mathcal{R}'$:= `Propagation_construction`(A, false, $\mathcal{R}$, `f_to_treat`, $\wp$`_waiting`);
if ($\mathcal{R}' \neq$ `failure`) then
$\mathcal{R}'$:= `Propagation_construction`(B, true, $\mathcal{R}'$, `f_to_treat`, $\wp$`_waiting`);
if ($\mathcal{R}' \neq$ `failure`) then
$\mathcal{R}'$:= `Search_free_leaf_to_treat`($\mathcal{R}'$, `f_to_treat`, $\wp$`_to_treat`, $\wp$`_waiting`);
(b) B := `Next_free_leaf` (B, dir);
(c) if $(B \neq A)$ then B := `Next_free_leaf` (B, dir);

Until $(B \neq A)$ and ($\mathcal{R}' \neq$ `failure`)

(iii) $\mathcal{R}$:= $\mathcal{R}'$.

Procedure `Propagation_construction` (A : sub-formula, `prop` : boolean, $\mathcal{R}$, variables `f_to_treat`, $\wp$`_waiting`) $\rightarrow$ `failure` or $\mathcal{R}$

(i) While B (the other premise of the link under the current sub-formula A) has been treated, do

(a) if it is a $\otimes$ link (for instance A$\otimes$B),
- if its premises belong to two disjoint elements of $\mathcal{R}$ then merge and extend them with the $\otimes$-link; add it to $\mathcal{R}$ where the two elements are erased; replace the current formula by the formula of this link (A := A$\otimes$B);
prop := true;
- else return(`failure`).

(b) if it is a $\wp$-link (for instance A$\wp$B),
- if its premises belong to the same element of $\mathcal{R}$ then the link is added to this subnet to give a new element of $\mathcal{R}$ where the initial element is erased; replace the current formula by the formula of this link (A := A$\wp$B) ;
prop := true;
- else the link is added to $\wp$`_waiting`.

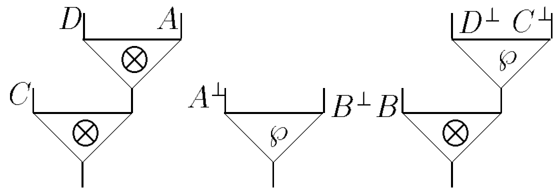
(ii) if (`prop` = true) and the link under the current sub-formula is a $\otimes$-link then its other premise is added at the end of the list `f_to_treat`.

(iii) Return($\mathcal{R}$).

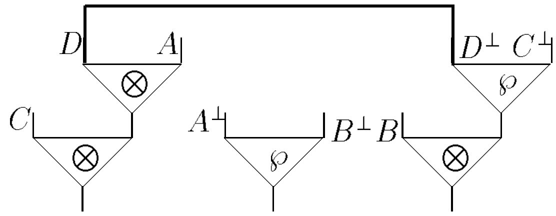
This procedure can be considered as a new method for automatic theorem proving in non-commutative linear logic, that is based on proof nets. Let us illustrate this algorithm with the sequents we considered in the previous section.

Example 1.

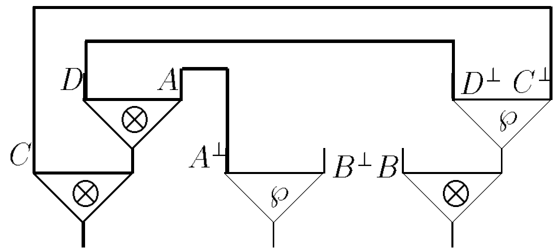
Let us start with the sequent $\vdash C \otimes (D \otimes A), A^{\perp} \wp B^{\perp}, B \otimes (D^{\perp} \wp C^{\perp})$. Its decomposition tree is the following:



We assume that the first leaf to treat is $D^{\perp}$. In this case, the search of the next free leaf to treat is done to the right, because this atomic formula is at the right of the nearest $\otimes$ link $(B \otimes (D^{\perp} \wp C^{\perp}))$.
Then the free leaves to consider[5] are $C^{\perp}$, D, $A^{\perp}$ and B and we can construct an axiom-link between $D^{\perp}$ and D.

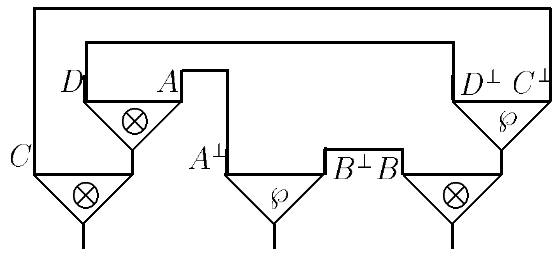


The next formula to treat is A with a next traversal to the right; and one has to find its dual formula among $A^{\perp}$ and B. At the next step, the next free leaf from B is A. From now, we have no more choice to create the two last axiom-links.



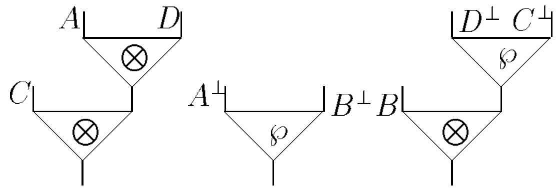
We can observe that the restrictions during the construction have imposed the condition of strong planity of the following final proof net.

[5] Let us recall that only the odd occurrences of the set of candidates are considered (cf section 4.3).



Example 2.

Let us consider now the sequent $\vdash C \otimes (A \otimes D), A^{\perp} \wp B^{\perp}, B \otimes (D^{\perp} \wp C^{\perp})$, that is the previous one where $D \otimes A$ is replaced by $A \otimes D$. Its decomposition tree is the following:



As in the previous construction, we assume that the first leaf to consider is $D^{\perp}$. The search of the next free leaf to treat is also done to the right, because this atomic formula is at the right of the nearest $\otimes$ link, namely $B \otimes (D^{\perp} \wp C^{\perp})$. The candidates as good free leaves are $C^{\perp}$, A, $A^{\perp}$ and B but we cannot construct an axiom-link from between $D^{\perp}$. Consequently, the construction algorithm stops with a failure and we deduce that the given sequent is not provable in non-commutative linear logic.

5 Properties of the algorithm

To prove the soundness and completeness of this algorithm we need some preliminary results on the planar graphs associated to this construction.

5.1 Preliminary results

Definition 5.1 Let G be a graph with, for each node, an associated cyclic permutation of its adjacent edges, G is a *planar map* if G is in a plan and the traversal of adjacent edges (at each node) is anti-clock-wise.

Let us remark that a MLL proof net to which is associated, at each node, the permutation (conclusion, right premise, left premise) is a planar map if and only if it is strongly planar.
We will give now a simple necessary and sufficient condition such that a connected graph remains a planar map when we add an edge to it.

Definition 5.2 Let G be a planar map and (s, a) with s node of G and a an edge adjacent to s, let $\tau(s, a) = $ (t, b) if $t \neq s$ and t be adjacent to a; if

$\sigma_t(b) = a$ (resp. $\sigma_t(a) = b$) [6], then the left (resp. right) *cell* of (s, a) is defined as the set $\{\tau^n(s, a)/n \in N\}$.

Lemma 5.3 *Let G be a planar map, s and t be two nodes of G, σ_s and σ_t be the associated cyclic permutations, let a and b be two edges respectively adjacent to s and t and let G' be the graph G to which one adds an edge u between a and b with the permutations σ_s and σ_t changed into σ'_s and et σ'_t with :*

$$\begin{cases} \sigma'_s(a) = u & \sigma'_t(b) = u \\ \sigma'_s(u) = \sigma_s(a) & \sigma'_t(u) = \sigma_t(b) \\ \sigma'_s(x) = \sigma_s(x) \text{ if } x \neq a, u & \sigma'_t(x) = \sigma_s(x) \text{ if } x \neq a, u \end{cases}$$

namely, u is added after a (resp. b) in the permutation.
If (t, b) (resp. $(s, \sigma_s(a))$) belongs to the left (resp. right) cell defined by (s, a) (resp. $(t, \sigma_t(b))$), then G' is a planar map.

Proof. Let us consider a strongly planar representation of G, it is sufficient to follow the path $(s, a), \tau(s, a), ..., (t, b)$ in such a way that we do not cross edges respecting the permutations in s and t. See illustration in figure 3. In the left graph, to link s and t it is sufficient to follow the path $(s, a), \tau(s, a), \tau^2(s, a) = (t, b)$ without crossing the edges. In the right one, if t does not belong to the left cell of (s, a) the added edge have to cross the left cell. □

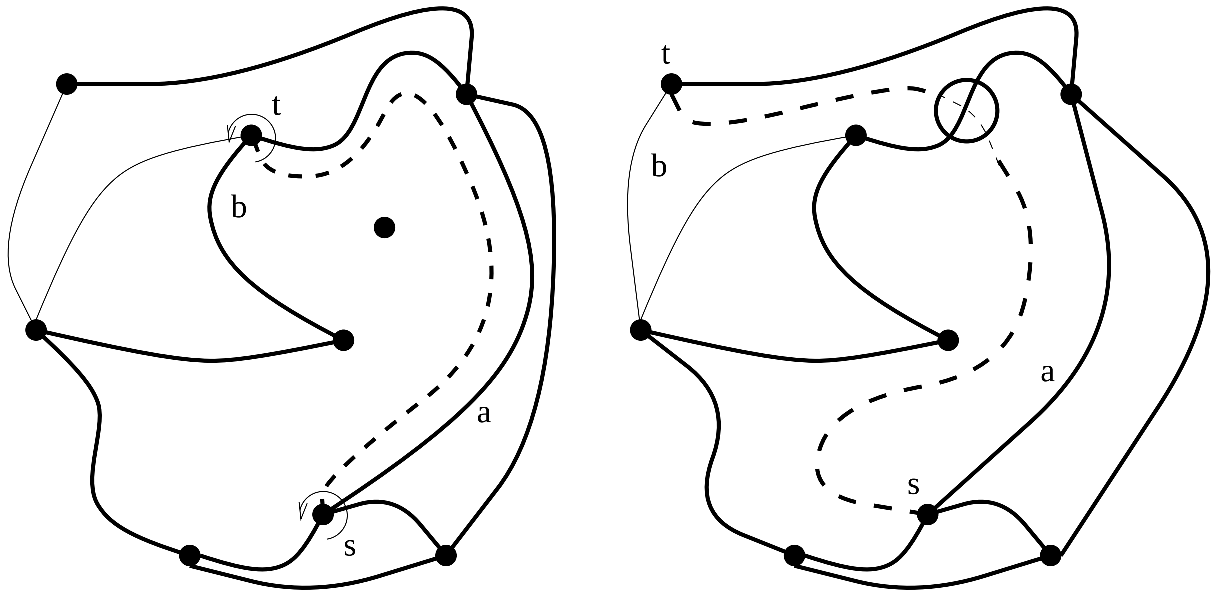


Fig. 3. Illustration of lemma 5.1

[6] σ_s and σ_t being the cyclic permutations on the adjacent edges of s and t.

Lemma 5.4 *Let us consider the same previous definitions, conversely, if G is connected then G' is a planar map if and only if (t, b) belongs to the left cell defined by (s, a).*

Proof. A left or right cell is a finite set (because G is finite) then the sequence $(\tau^i(s,a))_{i\in N}$ is cyclic from a certain rank. τ being a bijection (because σ is a bijection), $(\tau^i(s,a))_{i\in N}$ is then cyclic. Consequently a left or right cell determinates two parts of the plan.
As G is connected there is no node in the left (resp. right) part with the direction for traversal $\tau^0(s,a), \tau^1(s,a), ...$ for a left (resp. right) cell. The added edge is necessarily in this part to respect the permutation σ_s and then (t, b) belongs to the left cell of (s, a). □

5.2 Correctness and completeness

Let us consider now again the algorithm we have defined. The following result is important for the proof of correctness.

Lemma 5.5 *The iteration of the* `Next_free_leaf` *procedure on a free leaf A returns exactly the set of the free leaves such that the creation of an axiom-link between one of them and A preserves the strong planarity of the proof structure.*

Proof. Taking into account the symmetry, we consider that the traversal is done to the right. The atomic formulae are not directly represented in the planar maps corresponding to the proof structures but rather as pairs (s, a) defined as follows: s is the node representing the link just above the atomic formula. If the axiom-link starts from the formula then t is the edge of the graph representing it; if this formula is a left premise and if the right one is not atomic or if an axiom-link starts from it then t is the edge representing this premise; else t is the conclusion of s. Then, it is sufficient to verify that the `Next_free_leaf(.,` dir_R`)` procedure returns the next free leaf of the left cell defined by the free leaf on which it is called.
Moreover the `Next_atomic_formula(.,` dir_R`)` returns the next atomic formula of the left cell (that does not change during the construction). If an axiom link starts from this formula then its image by τ is the formula to which it is connected by this link. □

Theorem 5.6 *The algorithm for non-commutative proof net construction is* correct *(if, for a given sequent S, it returns a proof structure then S is provable in MNCLL) and* complete *(if S is a provable sequent in MNCLL then it returns a proof net of S).*

Proof. It is a direct consequence of the previous three lemmas and of the correctness and completeness results for the commutative proof nets construction algorithm (detailed in [7]). □

6 Concluding remarks

The main result here is the design of an algorithm for automated proof net construction in the non-commutative linear logic. From such a result we can also derive a procedure for automated deduction in this fragment. In fact, the algorithm can be directly used to construct simultaneously, step by step, the corresponding sequent proof following a top-down approach (from axioms to the sequent to prove). After the search of the axiom-links (corresponding to axioms of sequent calculus) a successful application of the `Propagation-construction` procedure corresponds to the application of adequate $\otimes$ or $\wp$ inference rules in a right order to lead to a proof.
Such a new algorithm and proof search method could be very helpful for applications in linguistics or distributed systems design where one has, at the same time, to deal with sequential and concurrent aspects. The relationships between non-commutative linear logic and the Lambek calculus [3] can lead to a new alternative method for Lambek proof nets construction and automated deduction without explicit use of λ-terms [17] or of labels on the links and of unification on these labels.

References

[1] V.M. Abrusci. Phase semantics and sequent calculus for pure noncommutative classical linear propositional logic. *Journal of Symbolic Logic*, 56(4):1403–1451, 1991.

[2] V.M. Abrusci. Noncommutative proof nets. In J.Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 271–296. Cambridge University Press, 1995.

[3] V.M. Abrusci. Lambek calculus, cyclic multiplicative-additive linear logic, noncommutative multiplicative-additive linear logic: language and sequent calculus. In V.M. Abrusci and C. Casadio, editors, *Proofs and Linguistic Categories - Proceedings 1996 Roma Workshop*, pages 21–48, 1996.

[4] A. Asperti and G. Dore. Yet another correctness criterion for multiplicative linear logic with mix. In *Logic at St Petersburg '94, Symposium on Logical Foundations of Computer Science, LNCS 813*, pages 34–46, St Petersburg, Russia, July 1994.

[5] G. Bellin and J. van de Wiele. Subnets of proof-nets in MLL^-. In J.Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 249–260. Cambridge University Press, 1995.

[6] V. Danos and L. Regnier. The structures of multiplicatives. *Arch. Math. Logic*, 28:181–203, 1989.

[7] D. Galmiche. Connection methods in Linear Logic and Proof nets Construction. *Theoretical Computer Science*, 1998/99. Accepted for publication.

[8] D. Galmiche and G. Perrier. A procedure for automatic proof nets construction. In *LPAR'92, International Conference on Logic Programming and Automated Reasoning, LNAI 624*, pages 42–53, St. Petersburg, Russia, July 1992.

[9] D. Galmiche and G. Perrier. Foundations of proof search strategies design in linear logic. In *Logic at St Petersburg '94, Symposium on Logical Foundations of Computer Science, LNCS 813*, pages 101–113, St Petersburg, Russia, July 1994.

[10] D. Galmiche and G. Perrier. On proof normalization in linear logic. *Theoretical Computer Science*, 135(1):67–110, 1994.

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

[12] J.Y. Girard. Towards a geometry of interaction. In J.W. Gray and A. Scedrov, editors, *Conference on Categories in Computer Science and Logic*, pages 69–108, Boulder, Colorado, June 1987.

[13] J.Y. Girard. Linear logic: its syntax and semantics. In J.Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 1–42. Cambridge University Press, 1995.

[14] J.Y. Girard. Proof nets: the parallel syntax for proof theory. In Ursini and Agliano, editors, *Logic and Algebra*, New York, 1995. M. Dekker.

[15] G. Gonthier, M. Abadi, and J.J. Levy. Linear logic without boxes. In *7th IEEE Symposium on Logic in Computer Science*, pages 223–234, Santa-Cruz, California, June 1992.

[16] P. Lincoln, J. Mitchell, A. Scedrov, and N. Shankar. Decision problems for propositional linear logic. *Annals of Pure and Applied Logic*, 56:239–311, 1992.

[17] G. Morrill. Clausal proofs and discontinuity. *L. J. of the IGPL*, 3(2,3):403–427, 1995.

[18] M. Nagayama and M. Okada. A graph-theoretic characterization theorem for multiplicative fragment of non-commutative linear logic - extended abstract. *Electronic Notes in Theoretical Computer Science*, 3:11 pages, 1996.

[19] M. Pentus. Equivalence of multiplicative fragments of cyclic linear logic and noncommutative linear logic. In *4th Int. Symposium LFCS'97, LNCS 1234*, pages 306–311, Yaroslavl, Russia, 1997.

[20] D. Roorda. Proof Nets for Lambek Calculus. *Journal of Logic and Computation*, 2(2):211–231, 1992.

[21] D.N. Yetter. Quantales and (noncommutative) linear logic. *Journal of Symbolic Logic*, 55(1):41–64, 1990.