Connection Methods in Linear Logic and Proof Nets Construction

D. Galmiche
LORIA - Université Henri Poincaré
Campus Scientifique - B.P. 239
54506 Vandœuvre-lès-Nancy Cedex, France.
e-mail : galmiche@loria.fr

Abstract

Linear logic (LL) is the logical foundation of some type-theoretic languages and also of environments for specification and theorem proving. In this paper, we analyse the relationships between the *proof net* notion of LL and the *connection* notion used for efficient proof-search in different logics. Aiming at using proof nets as a tool for automated deduction in linear logic, we define a connection-based characterization of provability in Multiplicative Linear Logic (MLL). We show that an algorithm for proof net construction can be seen as a proof-search connection method. This central result is illustrated with a specific algorithm that is able to construct, for a provable MLL sequent, a set of connections, a proof net and a sequent proof. From these results we expect to extend to other LL fragments, we analyse what happens with the additive connectives of LL by tackling the additive fragment in a similar way.

Keywords: linear logic, automated deduction, proof nets, connection methods.

1 Introduction

Linear Logic (LL), owing to its constructive nature and its symmetry, is often considered as a logic of computations [32, 40]. Many works on fragments of LL are devoted to programming, following the paradigms of *proofs-as-programs, proofs-as-computations* or *proof-search as computation*. In the first, the Curry-Howard correspondence leads us to consider the logic as a type-theoretic language and in the third the logic is, essentially, viewed as a logic of specification. In fact, LL is sufficiently refined to express many programming concepts and has an essential significance for the derivation of correct programs. In such works, the explicit construction of a proof, for a given sequent, appears either as a process of program development or as a process of computation; then usable and efficient methods for proof-search are necessary. Considering this proof-search (or construction) problem, we can mention two main points of view: adapting or extending to LL existing proof-search methods improved in classical logic (CL) or intuitionistic logic (IL) (as tableaux or connection methods) or developing more specific methods based on the proof-theoretical characteristics of LL (as for instance non-permutability results). In fact, some questions arise: can we consider, from a proof-theoretic point of view, LL only as a refinement of CL and IL and then enforce the use of various techniques usable in the other logics ? how could we take into account some notions and particularities of LL to define new techniques for proof-search ? is the problem to drastically reduce the search space of same nature in the different logics ?

There exists a variety of works on proof-search methods for Linear Logic and its fragments [29] for instance based on resolution [41], tableaux methods [19], permutability results or canonical proofs [25, 38]. All of them aim at drastically reducing the search space compared to calculi such as sequent calculi. The connection (or matrix) method [9, 48] is such a method, improved in CL and IL, that also avoids many redundancies during the proof-search. It could be interesting to define a based-on connection characterization of provability in some linear logic fragments, starting with the Multiplicative Linear Logic (MLL), and to propose corresponding connections proof-search methods. Focusing on this argument, to avoid many redundancies during the proof search as in sequent calculi, we can observe that using the notion of *proof net* could be a potential issue for proof-search. This notion, a new kind of syntax for linear logic, has been introduced by J. Y. Girard [27, 29] 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. It provides a nice, concise graph-theoretic representation of deductions in LL, eliminating most of the "syntactical bureaucracy" (cf. [27]) due to arbitrary sequentialization of inference rules. Let us recall that a *correctness criterion* is a procedure to identify proof nets (legal proofs) inside the universe of rough, ill-typed proofs (called *proof structures*). Works have provided different correctness criteria for proof structures of MLL or its extensions [5]. From an automated deduction point of view and also for an actual implementation of proof environments dealing with proof nets, we cannot only construct proof structures and verify *a posteriori* if they are proof nets but we also need procedures that directly build a proof net (or all the proof nets) of a given sequent. Let us mention that, starting from a first algorithm for automated proof net construction [24], we have conjectured relationships between proof nets construction and a possible based-on connection method for MLL [16].

The main ideas developed in this paper were first explicitly presented in 1996 (cf. [17]) and are issued from a more global study of the automated deduction in LL fragments using the *proof net* notion. Even if we could consider this proposal of connection characterization of provability in MLL as natural for specialists in the connection method, we show, as a main result, that an algorithm for automated proof nets construction [24, 20, 21], which could seem natural to specialists in proof nets, can provide a connection method for MLL and vice versa. As illustration, we consider a procedure for proof net construction developed in [24] and revised in [20]. An important characteristic of its derived connection method is that a sequent proof can be also constructed in parallel, avoiding problems such as sequent reconstruction from the two following points of view: construction of a sequent proof from a proof net [6] or construction of a sequent from the connection characterization (i.e. the subset of connections leading to provability) [44]. Independently, Kreitz and al. [35] have recently presented a characterization of logical validity in MLL and a connection-based proof-search procedure that is a straightforward extension of a method originally developed for classical and intuitionistic logics [42]. Among its features, let us mention the generality and uniformity but also the necessary use of a sequent reconstruction procedure. From our results, this method, initially dedicated to proof-search, could also be used for proof nets construction.

In fact, we emphasize the interest of the proof net concept, coming from the linear logic (and extended to other logics) of J.Y. Girard [27] as a tool for automated deduction. It has natural relationships with matrix or connection proof search methods (linear connection method [10] or connection method for intuitionistic logic [48]). Works on linear proofs (proofs in which each instance of a literal is connected at most once) and matrix have been developed [10, 15] from modifications of the connection method [4, 9] for instance in order to deal with planning problems. Linear logic has been independently studied since 1986 with powerful notions to deal with computations from both well-founded proof-theoretical and semantical points of view [27, 29].

The point is to consider formulae as resources and to manage them explicitly with creation and consumption of resources (which is already present in the Lambek's results). Lambek's work is on background of those ideas, but the history, the context, the evolution, the initial motivations and the applications of such approaches can clearly explain the lack of interaction between both, even if it was clear that these proof methods deal with concepts that are implicitly related to concepts of LL's proof theory. Let us notice that in 1984, [33] describes a decision procedure for direct logic, where contraction is eliminated, such that a formula can be used at most once in a proof. More recently, [8] has analysed its relationships with linear logic and in fact some concepts used in such procedures are implicitly related to concepts defined, or used, by computer scientists and logicians working on proof theory applications.

In this paper, we propose a new approach to define connection proof-search methods for linear logic fragments, using the notion of proof net as the central tool [30]. After recalling the various concepts and definitions, we propose a connection method that can be used to prove validity but that also simultaneously, and automatically, constructs both a proof net and a corresponding sequent proof. Such a method, implemented in interactive or automated theorem provers for LL, will produce some adequate and readable proof representations, compared to the traditional compact proof representation in connection methods. It is an important step of a more ambitious aim, to have adequate proof construction methods for computation in type-theoretic, logical and concurrent languages based on linear logic [34].
First, we study the multiplicative fragment of LL (MLL) for which the concept of proof net is well defined [30]. We establish a characterization of provability with a notion of spanning set of connections (or equivalently connection net). The design of a corresponding usable connection proof-search method for MLL is then based on an algorithm that constructs a proof net from a given sequent [24, 20, 21]. More precisely, the search for adequate connections for a given sequent is directed by the construction of a proof net for such a sequent. The main result is that a procedure for MLL proof nets construction can be seen as a based-on connection proof-search method. We illustrate this result considering an algorithm for proof nets construction firstly presented in [24] and recently revised with B. Martin for some extensions to other fragments [20, 21, 22]. With the derived connection method, we are able to avoid the sequent reconstruction from a proof net [6] or from the connection characterization (i.e., the subset of connections leading to provability) [44]. A dual result is that a connection method for MLL, for instance like the one of [35], can be used to construct proof nets. Further works could analyse such an approach from complexity and algorithmic points of view.
In fact, our approach is important for an analysis of computation (including failure) w.r.t. an initial logical specification. From these results in the MLL fragment the question of a possible adaptation to fragments including the additive connectives naturally arises. In general, these connectives are interesting for different reasons [28] but involve some problems due to the context dependency. It seems necessary to define specific techniques to include them into existing procedures for theorems proving in linear logic. Before taking into account the Multiplicative and Additive Linear Logic (MALL), we focus just on the Additive Linear Logic (ALL), for which a polynomial time semantic proof-search method in polynomial time has been defined [19]. Studying this fragment, we want to understand the special nature of the additive connectives during the proof search process. We thus propose a connection-based characterization for ALL that is a good illustration of some requirements to satisfy for such an approach. We also consider the relationships with the corresponding sequent proofs and proof nets within this fragment. We do not consider here the MALL case, but a recent proposal with B. Martin for an algorithm for automated MALL proof nets construction [21] could be a first step towards the design of a

similar proof-search method. From this work, we can think that we could potentially use the proof nets construction as an efficient technique for automated deduction in other LL fragments that are logical foundations of type-theoretic languages.
In fact, we also have explicitly established the relationships between concepts of proof-search (such as path, formula tree or connection) and logical concepts related to the proof net notion (as axiom-link, decomposition tree, proof structure). Such correspondences were often considered implicitly in works devoted to proof construction in direct logic [33] or linear logic [7, 8, 24] and the presentation of the relationships between connections and proof nets provides an opportunity to make them more explicit, from both the automated proof-search and the logical points of view.

2 Linear logic and proofs

The linear logic (LL) introduced by J.Y. Girard [27] is a logic of actions and resources. The main definitions, concepts and results about LL are given in [29] and a first overview of its applications in computer science is presented in [2]. We briefly recall the language (and the different logical fragments) and the linear one-sided sequent calculus is presented in Figure 1.

2.1 Linear Logic

The multiplicative fragment of propositional linear logic (MLL) is based on the constants 1 and $\perp$, the unary connective $()^\perp$ (*linear negation*) and the binary connectives $\otimes$ (*times*) and $\wp$ (*par*). The definition abbreviations are the following: $A^{\perp\perp} = A, 1^\perp = \perp, \perp^\perp = 1, (A \otimes B)^\perp = A^\perp \wp B^\perp, (A \wp B)^\perp = A^\perp \otimes B^\perp$. Let us notice that the *linear implication* can be defined as $A \multimap B \equiv A^\perp \wp B$. The MALL fragment is the extension of the previous one with the constants $\top$ and 0 and the additive connectives & (*with*) and $\oplus$ (*plus*) and the abbreviations: $\top^\perp = 0, 0^\perp = \top, (A \& B)^\perp = A^\perp \oplus B^\perp, (A \oplus B)^\perp = A^\perp \& B^\perp$. The fragment with exponentials is obtained by adding the unary connectives ! (*of course !*) and ? (*why not ?*) with: $(!A)^\perp = ?A^\perp$ and $(?A)^\perp = !A^\perp$. Finally the full linear logic LL includes the quantifiers $\forall$ (*every*) and $\exists$ (*some*) with: $(\forall x A)^\perp = \exists x A^\perp$ and $(\exists x A)^\perp = \forall x A^\perp$.

Let us recall that LL has also a two-sided sequent calculus having inference rules for negation. Because of the negation properties we can write $\Gamma \vdash \Delta$ as $\vdash \Gamma^\perp, \Delta$ with negation at the atom level. LL has a rich and well-developed proof theory that could be directly applied to have computational interpretations of the logic [2]. Works have been devoted to automated proof construction in various fragments and sequent calculi of LL [3, 25]. A way to improve the inherent difficulty of dealing with proof-search in sequent calculus (and the "bureaucracy of taxonomy" [27]) consists in using the notion of *proof net*, a graph with nodes marked by formulae and edges linking the premises of a rule to the conclusion, that was initially defined for the MLL fragment [27].
This fragment has been previously studied from the proof-search point of view but also for automated proof net construction [24, 20, 21]. To improve the efficiency of proof construction, we want to study the concept of connection-based proof method for MLL, knowing that such a method can be powerful [48]. First, we aim at giving a presentation of some definitions and results on MLL that emphasizes important relationships.

$$\frac{}{\vdash A, A^{\perp}}\ id \qquad \frac{\vdash \Gamma, A \quad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}\ cut \qquad \frac{\vdash \Gamma, B, A, \Delta}{\vdash \Gamma, A, B, \Delta}\ ex$$

$$\frac{\vdash \Gamma}{\vdash \Gamma, \perp}\ \perp \qquad \frac{}{\vdash \top, \Delta}\ \top_{ax} \qquad \frac{}{\vdash \mathbf{1}}\ \mathbf{1}_{ax}$$

Multiplicative rules

$$\frac{\vdash \Gamma_1, A_1 \quad \vdash A_2, \Gamma_2}{\vdash \Gamma_1, A_1 \otimes A_2, \Gamma_2}\ \otimes \qquad \frac{\vdash \Gamma, A_1, A_2}{\vdash \Gamma, A_1 \wp A_2}\ \wp$$

Additive rules

$$\frac{\vdash \Gamma, A_1 \quad \vdash \Gamma, A_2}{\vdash \Gamma, A_1 \& A_2}\ \& \qquad \frac{\vdash \Gamma, A_1}{\vdash \Gamma, A_1 \oplus A_2}\ \oplus_1 \qquad \frac{\vdash \Gamma, A_2}{\vdash \Gamma, A_1 \oplus A_2}\ \oplus_2$$

Exponential rules

$$\frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\ ! \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\ d? \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A}\ w? \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\ c?$$

Figure 1: A sequent calculus for propositional LL

2.2 A decision procedure for MLL

The following procedure is based on the search for axioms (or initial sequents) of a MLL sequent proof, that are in fact pairs of atomic formulae appearing negatively and positively in the sequent. It is a direct adaptation to MLL of previous works on direct predicate calculus [33] and on direct logic and linear logic [8]. The latter gives correspondences between basic data structures used in natural deduction (proof trees), linear logic (proof nets) and direct logic (chains). The interest of this presentation is that it involves concepts that are implicitly related to the notions of *proof structure* and *proof net* in linear logic. Even if there exists various improved proof-search algorithms for LL fragments we consider here an approach based on the construction of *paths* because of the relationships with the notions of *connections* [9, 48] and of *proof nets* [27].

The main notions and results are direct adaptations of [8, 14]. In order to distinguish the occurrences of atomic formulae we can extend the language with annotations on atomic formulae and then a formula is said to be *separated* if two occurrences of the same formula are differently annotated. We use the notation $A \prec B$ for A *occurs in* B (or A *subformula of* B).

The notion of *positive* and *negative occurrence* of a subformula is defined as usual: a formula A occurs positively in A and if A occurs positively (resp. negatively) in B then A occurs positively (resp. negatively) in $B \otimes C$, $C \otimes B$, $B \wp C$, $C \wp B$ and negatively (resp. positively) in $B^{\perp}$.

A *conjunctive subformula* $A \odot B$ is a positive occurrence of a conjunctive formula $A \otimes B$ or a negative occurrence of a disjunctive formula $A \wp B$.

Definition 2.1 *Let $\mathcal{P}$ be a set of pairs of atomic subformulae of a MLL sequent S;*
(i) $\mathcal{P}$ satisfies a formula *A (notation $\mathcal{P} \mapsto A$) if there exists a pair (P, P') in $\mathcal{P}$ such that either $P \prec A$ or $P' \prec A$.*
(ii) Let A, B be subformulae of S, A and B are connected in *$\mathcal{P}$ (notation $A \parallel_{\mathcal{P}} B$) if there is a pair (P, P') in $\mathcal{P}$ such that $P \prec A$ and $P' \prec B$ or vice versa.*
(iii) $\mathcal{P}$ has a conjunctive cycle *if there exists distinct conjunctive subformulae $A_0 \odot B_0, .., A_n \odot B_n$ of S such that $\forall i \in \{0, ..., n\}$ $B_i \parallel_{\mathcal{P}} A_{i+1}$, with indices considered modulo n.*

Definition 2.2 *(i) A* path $\mathcal{P}$ for a sequent S *is a set of pairs of atomic formulae such that*

a) if (P, P') *and* (Q, Q') *are distinct elements of* $\mathcal{P}$ *then* $P \neq Q$ *and* $P' \neq Q'$*;*

b) $\mathcal{P}$ *satisfies a formula of* S*;*

c) if $(P, P') \in \mathcal{P}$ *then* P *occurs positively in* S *and* P' *occurs negatively in* S*;*

d) if $\mathcal{P}$ *satisfies a conjunctive subformula* $A \odot B$ *then* $\mathcal{P}$ *satisfies* A *and* $\mathcal{P}$ *satisfies* B*;*

(ii) A path $\mathcal{P}$ *for a sequent* S *is* minimal *if no proper subset of* $\mathcal{P}$ *is a path for* S*.*

Definition 2.3 *The* path $\mathcal{P}_{\mathcal{D}}$ *of a derivation* $\mathcal{D}$ *is the set of the axioms (or initial) sequents occurring in* $\mathcal{D}$*.*

We can thus extend the previous definitions of satisfied and connected formulae to $\mathcal{P}_{\mathcal{D}}$.

Definition 2.4 *A derivation* $\mathcal{D}$ *is a* satisfied derivation *if we have*
i) the active formulae of each two-premise rule in $\mathcal{D}$ *are satisfied in* $\mathcal{D}$*.*
ii) at least one of the active formulae of each one-premise rule in $\mathcal{D}$ *is satisfied in* $\mathcal{D}$*.*

Proposition 2.1 *(i) If* S *is a provable sequent then there exists a satisfied derivation* $\mathcal{D}$ *of* S*.*
(ii) If $\mathcal{D}$ *is a satisfied derivation of* S *then* $\mathcal{P}_{\mathcal{D}}$ *is a minimal path for* S*.*

The proofs of both propositions are direct adaptations of [8, 14].

Theorem 2.1 *A MLL sequent* S *is provable if and only if there exists a minimal path* $\mathcal{P}$ *for* S *satisfying the two following conditions*

(α) if $\mathcal{P}$ *satisfies a conjunctive subformula* $A \odot B$ *then* A *and* B *are not connected in* $\mathcal{P}$*.*

(β) there is no conjunctive cycle in $\mathcal{P}$*.*

Proof 2.1 *The "only if" is proved from the previous proposition knowing that the path corresponds exactly to the axioms of the proof. As far as the "if" is concerned, given a minimal path for* S*, we have to construct a proof of it. The proof is similar to that of [8].*

Thus the decision problem can be reduced in this case to the search for a specific minimal path. Even if this process is exponentially bounded, the actual time is determined by the search process. To find a path for a sequent S, we have to design a procedure taking into account techniques to eliminate irrelevant information.

2.3 A proof-search procedure

From the previous *path characterization* of MLL provability, we can try to define a proof-search procedure, that constructs a minimal path satisfying the conditions of theorem 2.1.
A first refinement in this direction is the analysis of the construction of proofs for a sequent S from a path. It can be seen also as a process of sequent reconstruction [44]. That is why we directly adapt the results of [14] about Direct Logic with the following definitions:

Definition 2.5 *Let* S *be a sequent and* $\mathcal{D}$ *be a derivation of* S*; a conjunction subformula* $A \odot B$ *is a* path conjunction *if we can split* S *into* $S_1, S_2, A \odot B$ *and* $\mathcal{P}_{\mathcal{D}}$ *into* $\mathcal{P}_1$ *and* $\mathcal{P}_2$ *such that* $\mathcal{P}_1$ *is a path for* S_1, A *and* $\mathcal{P}_2$ *is a path for* S_2, B*.*

Definition 2.6 *(canonical proof)*
A proof $\mathcal{D}$ *of the sequent* S *is* canonical *if*
(1) either it is an axiom;

(2) or $\mathcal{D}$ has the form $\dfrac{\begin{matrix}\mathcal{D}_1 & \mathcal{D}_2\\ \vdots & \vdots \\ S_1 & S_2\end{matrix}}{S}\ R$ *or* $\dfrac{\begin{matrix}\mathcal{D}_1\\ \vdots \\ S_1\end{matrix}}{S}\ R$ *with $\mathcal{D}_1$ and $\mathcal{D}_2$ canonical and*

- *if S contains a disjunction ($\wp$) then R is the $\wp$ rule,*
- *or else if S contains a path conjunction then R is the $\otimes$ rule.*

It is easy to prove that any proof of a MLL sequent is equivalent to a canonical proof, mainly because of permutability results in this logical fragment.
Let us consider $\mathcal{P}$ as a path for the sequent S. We can construct a canonical proof of it (by induction on the size of S) as follows:

- if there is an axiom in S then we apply the *axiom* rule;
- or else if S contains a disjunction ($\wp$) then we apply the corresponding rule;
- or else if S contains a path conjunction ($\otimes$) then we apply the corresponding rule;
- or else return failure.

We can easily verify that the resulting proof (notation: $cp(\mathcal{P})$), if there exists, is canonical. Moreover we have the following result:

Proposition 2.2 *If S is provable in MLL then there exists a path $\mathcal{P}$ for S such that $cp(\mathcal{P})$ is defined and is a proof of S.*

Proof 2.2 *the proof is similar to that of [14].*

Thus we have another characterization based on the existence of a path connected to this notion of canonical proof, but no direct algorithm to have directly build a good path.
About this notion of canonical proof, we have to mention that we have previously defined such a notion for different fragments of LL as MLL, MALL and also for full linear logic (CLL) but without reference to an initial path [23, 26]. Moreover, this class of proofs is complete w.r.t. the provability and then a simple proof-search procedure consists in, from an initial sequent, trying to build such a specific proof. To summarize, we have presented a characterization of provability based on the notion of path (set of pairs of atomic formulae) and connected it with the construction of particular sequent calculus proofs.

3 Linear logic and Proof nets

The notion of *proof net*, a new kind of syntax for linear logic, has been introduced by J.Y. Girard [27, 29] 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. Proof nets provide a concise graph-theoretic representation of deductions of linear logic, eliminating most of the "syntactical bureaucracy" [27] due to an arbitrary sequentialization of inference rules[1]. The notion of proof net, originally defined for MLL, has been defined or extended to other fragments such as MALL or non-commutative linear logic [1, 29, 31]. Taking into account the sequentialization problem, a main problem consists in finding the most adequate or natural definition of proof net in a given

[1] We consider Multiplicative Linear Logic (MLL) in which we ignore the role of the units and thus the fragment called here MLL is in fact MLL^-

logical fragment. The relation between premises and conclusions of *links* induces a partial order on the so-called *proof structures* that coincides with the relation of "*being a subformula*". In fact only the axiom-links are needed to define a proof structure once a tree of subformulae is given. A proof structure is a set of formula occurrences connected by certain relations (links) and proof nets are the proof structures that represent correct linear proofs.

3.1 Proof net definitions

Among the different proof nets definitions we consider first the one of [8] that is closer to the previous notion of path. For that, for a given MLL sequent S, we consider two sets:
(a) the set $S^{\prec}$ of its subformulae of arranged as the decomposition tree where a link is the relation between a formula (*conclusion* of a link) and its immediate subformulae (*premises* of a link);
(b) a set $\mathcal{P}$ of axiom-links (links between positive and negative occurrences of the same atomic formula). Each occurrence of atomic formula in $S^{\prec}$ is a conclusion of at most one axiom-link.

Definition 3.1 *A pair* $(S^{\prec}, \mathcal{P})$ *is a* proof structure *for MLL if each atom of* $S^{\prec}$ *is the conclusion of exactly one axiom link.*

A proof net is a proof structure that satisfies a graph-theoretic condition that represents certain consistency conditions on the structure as a whole [7]. For instance, in [27] such condition is defined in terms of trips over $(S^{\prec}, \mathcal{P})$ that visit the formulae of a proof structure in two directions, the movements being determined by the nature of the links and by arbitrary choices (switches). In [33] where the decision procedure for direct logic contained implicitly a notion similar to the one of proof net, the global condition is acyclicity, where a cycle is defined as a chain conjunctive subformulae satisfying certain properties. We consider here the so-called Danos-Regnier switching condition that is related to the previous one.

Definition 3.2 *A* Danos-Regnier switching *s (DR-switching) in a proof structure corresponds to the choice, for each $\wp$-link, of one of the premises of the links.*

Definition 3.3 *Let* $\mathcal{PS}$ *be a proof structure and s be a switching, we define a* Danos-Regnier graph *(DR-graph* $\mathcal{PS}_s$*) as follows:*
(i) the vertices of $\mathcal{PS}_s$ *are the formulas of* $\mathcal{PS}$*;*
(ii) there is an edge between vertices X *and* Y *exactly when:*
a) X *and* Y *are conclusions of an axiom link or the premises of a cut link;*
b) X *is a premise and* Y *the conclusion of a* $\otimes$ *link;*
c) Y *is the conclusion of a* $\wp$ *link and* X *is the occurrence selected by the switching* s*.*

Definition 3.4 *A proof structure* $(S^{\prec}, \mathcal{P})$ *is a* proof net *for MLL if for every DR-switching s in this proof structure then the DR-graph is acyclic and connected.*

The following fundamental result relates provability (and sequent calculus) and proof nets in MLL [27].

Theorem 3.1 *S is provable in MLL if and only if there exists a set of axiom-links* $\mathcal{P}$ *such that* $(S^{\prec}, \mathcal{P})$ *is a proof net.*

Proof 3.1 *See [11].*

This result gives another way of characterizing the MLL provability through the existence of a proof net and more precisely the existence of a set of so-called axioms-links. The relationships between such a set and a path (as previously defined) are important to understand the connections between proof nets and proof-search algorithms in the based-on matrix methods [4, 9, 48] and also why some decision procedures are implicitly based on structures corresponding to what is called proof nets in linear logic. Let us recall that there exists a polynomial time algorithm that decides whether a proof structure is a proof net or not and that the decision problem for MLL is NP-complete [37].
To complete this presentation, taking into account that our aim is to have efficient algorithms to construct directly and automatically proof nets, it appears interesting to use an inductive definition of MLL proof nets (similar to the one proposed in [24]). It is given in appendix A.

3.2 Proof nets characterization

Let us recall that there exists a bijection between proof nets and the equivalence classes of sequent derivation modulo permutation of inferences. Given a proof net we have a polynomial time method to obtain a corresponding sequent derivation (sequentialization theorem).
To complete the previous presentation we emphasize here that there is an equivalence between the previous decision procedure and the construction of proof nets following their equivalence with the sequent calculus.

Proposition 3.1 *If Π is a MLL proof net (without cut-links) with S as a set of conclusions in MLL then there exists a minimal path $\mathcal{P}$ for S that satisfies the conditions (α) and (β) of theorem 2.1.*

Proposition 3.2 *If a proof structure $(S^{\prec}, \mathcal{P})$ is such that $\mathcal{P}$ satisfies the conditions (α) and (β) of theorem 2.1 then it is a proof net Π.*

The proofs of these propositions are similar to the ones of [8] and a direct graph-theoretic proof of this equivalence can be also proposed. Moreover, these results justify that an alternative proof-search method could be based on proof nets construction.

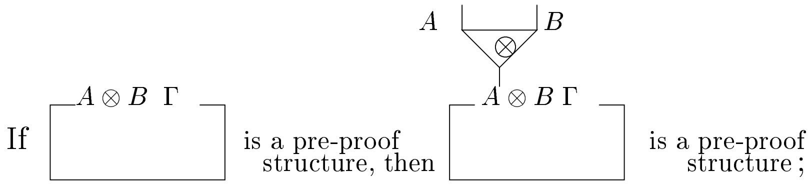
3.3 Proof nets construction

There are various equivalent criteria to define a proof net from a proof structure. A first approach for proof net construction consists, for a given sequent, in constructing a proof structure and in verifying afterwards with one of these correctness criteria if the structure is finally a proof net. In fact, it corresponds more to proof net verification for which we know that there exists a polynomial time algorithm that decides whether a proof structure is a proof net or not. Another one consists in a direct construction of a proof net if it exists. In this context, it is debatable whether the calculus of proof nets is a perspicuous method of proof if the verification of any of the different consistency conditions is part of the formal representation of the deductive system. For instance it seems that "acyclicity condition" [33] is more perspicuous than the "trip condition" [27]. A proof net can be then regarded as the result of an inductive inferential process as formalized in a sequent calculus derivation. Hence, a proof net would always be given together with some derivation in the sequent calculus (or other global inductive process) [8]. To consider this construction way, we will implicitly use the inductive definition of MLL proof nets. A first algorithm for proof nets construction based on these principles has been presented in [24] and more recently a refined version has been proposed in [20, 21]. The later has been used as starting

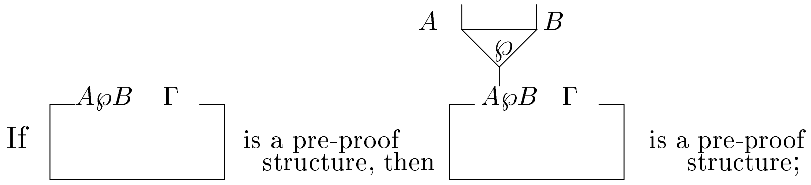
point for proof nets construction algorithms for the extension to the additives (MALL) [20, 21] and for the non-commutative MLL fragment [22]. Our aim here is to consider proof nets calculus as a basis of proof-search methods. Moreover, if we consider the previous relationships, the proof net construction corresponds also to the direct computation of a set $\mathcal{P}$ of axiom-links having the good properties. As we will see in the next section, such a set corresponds exactly to the set of so-called connections used in the proof-search methods based on the matrix (or connection) characterizations of provability [4, 9].
Consequently, we will focus on this structure, namely the proof net, explicitely defined by Girard [27] but implicitly present, through different notions, in works on proof-search in resource-aware logics like direct logic [33]. Then we will consider a new algorithm for proof net construction as an efficient tool for a proof search method. Moreover, if we want to use the proof net notion as a basis of some programming paradigms we need efficient methods that directly return, from a given sequent, a proof net, if it exists.
We present now some basic and simple notions that are useful for such methods. They include some inductive definitions as the following:

Definition 3.5 *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:*

1. *A sequence $A_1, ..., A_n$ is a pre-proof structure with $A_1, ..., A_n$ as leaves.*
2. *$\otimes$-link:*



3. *$\wp$-link:*



In fact, for a sequent S, the set $S^{\prec}$ naturally corresponds to the decomposition tree structure inductively defined as follows:

Definition 3.6 *A pre-proof structure of S, the leaves of which are literals, is called the* decomposition tree *of S (*terminal branch*).*
A structure under construction *of S is the decomposition tree of S where some leaves, labelled by dual literals, are linked with an axiom-link.*
A free leaf *of a structure under construction Θ of S is a leaf of the decomposition tree that is not the extremity of an axiom-link.*

Thus 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 and then an adequate set $\mathcal{P}$. Let us recall that regarding the cut elimination result [27] we will consider here proof nets without cuts.
After discussing about decision procedures and proof nets in MLL, we have to analyse what a connection- or matrix-based characterization of provability in MLL could be, having in mind that the proof search methods using such characterizations (in classical or intuitionistic logics) manipulate structures that are implicitly similar to the so-called proof nets.

4 A connection characterization for MLL

If we consider proof-search in sequent calculus, for example in classical logic but also in some LL fragments, three major redundancies are often identified within the search space [48], namely, *notational redundancy* (duplication of information), *irrelevance* (considering reductions that do not advance the search towards finding a proof) and *non-permutability* (distinguishing derivations that differ in the order of the sequent rules application). Faced with such problems, proposals called *matrix (or connection) characterizations* of the logic have been developed and improved [9, 48] to study provability in classical and intuitionistic logics. The advantage of such a method is that the direct search of so-called connections drastically reduces the search space compared to sequent or tableaux calculi.
Then it seems natural to analyse whether such an approach could be defined in linear logic and then improve the works on proof-search in linear sequent calculus [25, 46]. Moreover, we also remark that the notion of proof net, firstly defined in MLL, can be seen as a possible solution to the major above-mentioned redundancies. Consequently, we propose in the next sections a connection characterization for the MLL fragment of linear logic and then we analyse its relationships with the proof nets construction problem. We have conjectured such relationships in [16] from the initial algorithm for proof nets construction, developed with G. Perrier [24]. Our general work about proof nets as tools for automated deduction naturally led us to the ideas, concepts and results developed in this paper, that were initially presented in 1996 [17]. Let us notice that another work on connection proof-search method for linear logic has been independently proposed by Kreitz and al. [35], as a straightforward extension of a method originally developed for classical and intuitionistic logics [42]. Generality and uniformity are its main features.

4.1 Definitions and concepts

It seems natural to keep if possible the notions and definitions (adapted for the linear logic) used in [48] for connection characterization for classical logic. As a MLL sequent of the form $A_1, A_2, ..., A_n \vdash B_1, B_2, ..., B_m$ is interpreted as $A_1 \otimes A_2 .... \otimes A_n \vdash B_1 \wp B_2 .... \wp B_m$ the adaptations of the concepts dedicated to classical logic are direct. The goal consists in characterizing provability with conditions on sets of so-called connections as in the classical case but in MLL we need some modifications to take into account the different occurrences of literals and we have only to consider some suitable subsets of the set of all the connections.

Definition 4.1 *A* signed formula *is a pair $< A, n >$ where A is a formula and $n \in \{0, 1\}$.*
It is called atomic *if its constituent formula is atomic.*
A signed formula of the form $< A \otimes B, 1 >$, $< A \wp B, 0 >$, $< A^{\perp}, 1 >$ or $< A^{\perp}, 0 >$ is of conjunctive type *(or α-type). We define the components α_1, α_2 as follows:*

α	α_1	α_2
$< A \otimes B, 1 >$	$< A, 1 >$	$< B, 1 >$
$< A \wp B, 0 >$	$< A, 0 >$	$< B, 0 >$
$< A^{\perp}, 1 >$	$< A, 0 >$	$< A, 0 >$
$< A^{\perp}, 0 >$	$< A, 1 >$	$< A, 1 >$

A signed formula of the form $< A \otimes B, 0 >$ *or* $< A \wp B, 1 >$ *is of* disjunctive type *(or β-type). We define the components β_1, β_2 as follows:*

β	β_1	β_2
$< A \otimes B, 0 >$	$< A, 0 >$	$< B, 0 >$
$< A \wp B, 1 >$	$< A, 1 >$	$< B, 1 >$

For this presentation we consider one-sided MLL sequents of the form $\vdash A_1, A_2, ..., A_p$ that have $A_1 \wp A_2 .. \wp A_p$ as corresponding formula. Then we use a structure called *formula tree* which contains a name (or position) for each subformula of the endsequent.

Definition 4.2 *A* formula tree $\mathcal{FT}$ *is a representation of a sequent S (or its associated formula $\mathcal{F}$) as a tree, the nodes of which are marked by* positions *that correspond to* labels.
The tree ordering *is defined as the semi-subformula ordering: a position k is below a position k' (notation: $k \ll k'$) if the formula associated with k' is a proper semi-subformula of the formula associated with k. Moreover* lab(k) *denotes the subformula associated with k.*

Each position has a polarity, a principal type and a secondary type. The *polarity* is defined from the label and the polarity of its parent, the *principal type* of a position is defined from its polarity and its label and the *secondary type* from the principal type of its parent.

The example. To illustrate these notions we consider an example that consists in proving the following MLL sequent: $\vdash (A \otimes B) \wp ((A^{\perp} \wp B^{\perp}) \otimes A) \wp (A^{\perp} \otimes B) \wp B^{\perp}$.
A first step consists in constructing the formula tree from the endsequent with the polarity 0 (including labels, polarities denoted 0 or 1 and the principal and secondary types). It is given in figure 2 that includes all main necessary information. Now let us recall that when extending a derivation by reducing one of its leaves we have to decide which subformula of the leaf to reduce. There is no criteria for such choices and some works proposed obvious strategies for this disjunctive choice. Andrews [4] and Bibel [9] have proposed methods based on the notion of a *path through* a sequent.

Definition 4.3 *(i) Let S be a sequent, a* path through S *(or the associated formula $\mathcal{F}$) is a subset of the positions of its formula tree defined as follows:*
The set of paths through S is the smallest set such that:

1. *$\{a_0\}$ is a path, where a_0 is the root position of the formula tree for S;*
2. *if s, α is a path then $(s \setminus \{\alpha\}), \alpha_1, \alpha_2$ is a path;*
3. *if s, β is a path then $(s \setminus \{\beta_1\})$ and $(s \setminus \{\beta_2\})$ are paths.*

(ii) An atomic path *through S is a path composed only by atomic positions.*

This notion of path is completely different from the one of section 2.1 (definition 2.2). Here a path through a sequent is a set of subformulas and not a set of pairs of atomic formulae. But we keep in both cases the initial names for these notions we have adapted to the LL case. We can say that the set of paths through S corresponds to the set of sequents we can construct from S. Moreover, the atomic paths are the set of leaves of a derivation that has been extended as far as possible by reducing non-atomic formulae.

u	pol(u)	lab(u)	Ptype(u)	Stype(u)
a_0	0	$(A \otimes B)\wp((A^\perp \wp B^\perp) \otimes A)\wp(A^\perp \otimes B)\wp B^\perp$	α	-
a_1	0	$A \otimes B$	β	α_1
a_2	0	A	-	β_1
a_3	0	B	-	β_2
a_4	0	$((A^\perp \wp B^\perp) \otimes A)\wp(A^\perp \otimes B)\wp B^\perp$	α	α_2
a_5	0	$((A^\perp \wp B^\perp) \otimes A)$	β	α_1
a_6	0	$A^\perp \wp B^\perp$	α	β_1
a_7	0	$A^\perp$	α	α_1
a'_7	1	A	-	α_1
a_8	0	$B^\perp$	α	α_2
a'_8	1	B	-	α_2
a_9	0	A	-	β_2
a_{10}	0	$(A^\perp \otimes B)\wp B^\perp$	α	α_2
a_{11}	0	$A^\perp \otimes B$	β	α_1
a_{12}	0	$A^\perp$	α	β_1
a'_{12}	1	A	-	α_1
a_{13}	0	B	-	β_1
a_{14}	0	$B^\perp$	α	α_2
a'_{14}	1	B	-	α_1

Figure 2: The formula tree of $(A \otimes B)\wp((A^\perp \wp B^\perp) \otimes A)\wp(A^\perp \otimes B)\wp B^\perp$.

Definition 4.4 *A* connection *is a pair of atomic positions* (P, P') *in some paths through* S*, the labels of which are identical but with different polarities.*

The example (continued). Let us come back to the example and construct the set of atomic paths through the sequent $S \equiv \vdash (A \otimes B) \wp ((A^\perp \wp B^\perp) \otimes A) \wp (A^\perp \otimes B) \wp B^\perp$.

Starting from the formula tree of figure 2 and applying the inductive definition of the set of paths through a sequent, we finally obtain the following atomic paths:

(1) $\{a_2, a_7, a_8, a_{12}, a_{14}\}$ including the connections (a_2, a_7), (a_2, a_{12}).
(2) $\{a_2, a_7, a_8, a_{13}, a_{14}\}$ including the connections (a_2, a_7), (a_8, a_{13}), (a_{13}, a_{14}).
(3) $\{a_2, a_9, a_{12}, a_{14}\}$ including the connections (a_2, a_{12}), (a_9, a_{12}).
(4) $\{a_2, a_9, a_{13}, a_{14}\}$ including the connection (a_{13}, a_{14}).
(5) $\{a_3, a_7, a_8, a_{12}, a_{14}\}$ including the connections (a_3, a_8), (a_3, a_{14}).
(6) $\{a_3, a_7, a_8, a_{13}, a_{14}\}$ including the connections (a_3, a_8), (a_3, a_{14}), (a_8, a_{13}), (a_{13}, a_{14}).
(7) $\{a_3, a_9, a_{12}, a_{14}\}$ including the connections (a_3, a_{14}), (a_9, a_{12}).
(8) $\{a_3, a_9, a_{13}, a_{14}\}$ including the connections (a_{13}, a_{14}), (a_3, a_{14}) .

Let us consider this sequent and the classical sequent obtained by replacing $\wp$ by $\vee$, $\otimes$ by $\wedge$ and $\perp$ by $\neg$. For such sequent, a set of connections (in fact complementary connection including some constraints see [48]) is said to span S if every atomic path contains a connection from the set. Then the connection characterization for provability in propositional classical logic corresponds to the existence of a set of connections that spans the initial sequent. Here the classical version of the example is then provable. Because LL is a logic taking into account the different occurrences of a formulae we have to adapt the notion of spanning to the linear case and then to deal with particular subsets of the set of connections.

4.2 Connection characterization for MLL

As said before, we adapt if possible the definitions used in [48] for the connection characterization to the linear case.

Definition 4.5 *A set $\mathcal{SC}$ of connections (linearly)* spans *the sequent S (or the formula $\mathcal{F}$) if*
(a) all atoms in the formula tree appear exactly once *with polarity* 0 *or* 1 *in $\mathcal{SC}$;*
(b) if (P, P') and (Q, Q') distinct elements of $\mathcal{SC}$ then $P \neq Q$ and $P' \neq Q'$;
(c) every atomic path contains a connection from $\mathcal{SC}$;
(d) it is minimal (no proper subset of $\mathcal{SC}$ spans S).

Another way to cover the same notion with a different structure with strong similarities with the notions of path for a sequent and of proof net (including set of axiom-links) is the following:

Definition 4.6 *A* MLL connection structure *for the sequent S (or for the associated formula $\mathcal{F}$) is a pair $(\mathcal{FT}, \mathcal{SC})$ where $\mathcal{FT}$ is the formula tree and $\mathcal{SC}$ is a subset of the set of all the connections such that all atoms in S appear in exactly one connection of $\mathcal{SC}$.*
A MLL connection structure $(\mathcal{FT}, \mathcal{SC})$ is complementary *if each atomic path through S contains a connection of $\mathcal{SC}$. A MLL connection structure $(\mathcal{FT}, \mathcal{SC})$ is* minimal *if the set $\mathcal{SC}$ is minimal, i.e., if we drop a connection then the complementarity property is lost.*

Definition 4.7 *A* MLL connection net *for the sequent S is a MLL connection structure $(\mathcal{FT}, \mathcal{SC})$ that is complementary and minimal.*

Before considering the main result that is the characterization of provability in MLL, we study the relationships between the previous notions that are in fact trivially equivalent.

Proposition 4.1 *Let S be a given sequent and $\mathcal{F}_{\mathcal{T}}(\mathcal{S})$ be the corresponding formula tree,*
(i) if $\mathcal{SC}$ is a set of connections that spans S then the pair $(\mathcal{F}_{\mathcal{T}}(\mathcal{F}), \mathcal{SC})$ is a connection net.
(ii) if $(\mathcal{FT}, \mathcal{SC})$ is a connection net for S than $\mathcal{SC}$ is a set of connections that spans S.

We now give a characterization of the existence of a proof for a given MLL sequent and hence of its validity in terms of the existence of a MLL connection net from the sequent.

Theorem 4.1 *A MLL sequent S is provable if and only if there exists a MLL connection net for S.*

Proof 4.1 *From theorem 2.1, a MLL sequent S is provable if and only if there is a minimal path $\mathcal{P}$ for S that satisfies the conditions (α) and (β). Then let us prove that there is a minimal path $\mathcal{P}$ for S satisfying the conditions (α) and (β) if and only if there exists a MLL connection net for S.*
- *Case "if".* Let us assume that there exists a MLL connection net $(\mathcal{FT}, \mathcal{SC})$ for S.
$\mathcal{SC}$ is a subset of the set of connections that is complementary and minimal. Taking into account that a position a_i (in a connection) corresponds to a formula $lab(a_i)$ with a polarity (0 or 1), we have a correspondence between the set $\mathcal{SC}$ and the set $\mathcal{SC}_l$ that is the set of the corresponding atomic formulae occurring positively or negatively (depending on the polarity) in the initial sequent. We can prove that a minimal set $\mathcal{SC}_l$ is a set of pairs of atomic formulas that verifies the conditions a), b), c), d) of definition 2.2.
Thus if $\mathcal{SC}$ is minimal then $\mathcal{SC}_l$ is a minimal path for the sequent.

Because of the definition of $\mathcal{SC}$ and from the rule for path construction for the conjunctive sub-formula with $\odot$ as connective, the condition (α) of theorem 2.1 is satisfied for $\mathcal{SC}_l$. Moreover, the condition (β) is satisfied because if not, the existence of a conjunctive cycle in $\mathcal{SC}_l$ would imply that $\mathcal{SC}$ would include at least an atomic path without connections.
- Case "only if". Let us assume that there exists a minimal path $\mathcal{P}$ satisfying both conditions and consider ($\mathcal{FT}, \mathcal{P}^c$) where $\mathcal{P}^c$ is a subset of connections corresponding to $\mathcal{P}$. We can prove that it satisfies the conditions to be a MLL connection net.

Let us complete this part with the characterization of the MLL provability in terms of the existence of a spanning set of connections for the initial sequent.

Theorem 4.2 *A MLL sequent S is provable if and only if there exists a set $\mathcal{SC}$ of connections that spans S.*

Proof 4.2 *Direct consequence of proposition 4.1 and theorem 4.1.*

Our aim is now to propose a proof-search method based on such a connection characterization that computes a set of connections spanning the initial sequent

4.3 Connections in MLL

Coming back to classical logic, a very inefficient proof method consists in checking that every atomic path through the formula contains a connection. Methods developed by Andrews [4] and Bibel [9] focus more on the search of connections as opposed to the search of paths. These connection (or matrix) methods first identify a connection and then eliminate paths that contain it. Thus the consideration of paths is driven by the identification of connection and not vice versa. Consequently it deals with the irrelevance problem.
A connection method for MLL is a proof-search method that will be based on the search of connections (linearly) spanning the given sequent from the sets of paths and included connections. In this case, as for classical or intuitionistic logic [42], the proof-search is driven by connections instead of connectives as in the sequent calculus. But to take into account the specificity of LL about the formulae-as-resources management, we have to refine the search of connections. In fact, we can describe such a proof-search as follows: if a connection (a_i, a_j) has been identified in an atomic path, this path is eliminated as well as all the connections including one of the positions a_i and a_j in all the other atomic paths.

The example (continued). Coming back to the example, we are able to propose the following set of connections $\mathcal{SC} \equiv \{(a_{13}, a_{14}), (a_3, a_8), (a_9, a_{12}), (a_2, a_7)\}$ that spans the initial sequent. Consequently the given sequent is provable.
But how can we compute such a set of connections ? An adaptation of the classical method we have just described consists in starting if possible with paths having only one connection. Thus in our example, the path (4) includes only one connection, i.e., (a_{13}, a_{14}). As all connections including the labels a_{13} and a_{14} are erased then we can deduce the connection (a_3, a_8) from the path (5). Going on, we can deduce the connection (a_9, a_{12}) from the path (7) and finally the connection (a_2, a_7) from path (2). As each set of connections will be finally empty after iteration of this process we can deduce that this set of connections spans the initial formula.

The connection method is efficient to find proofs in classical logic according to the characterization but the resulting connections set is difficult to understand or to manage with. Some attempts have been made to convert such proofs for instance into a sequent calculus form by specific reconstruction techniques [44]. We will notice that the relationships between connection method and proof net construction could be appropriate to propose a more readable representation of proofs.

4.4 Relating connection method and proof nets construction

J.Y. Girard has defined the notions of *proof structures* and *proof nets* as bases of natural deduction systems in LL [30] . If we consider the presentation of section 3 and the problem of proof net construction for a given sequent S, the set $S^{\prec}$ of the subformulae of S is arranged as the decomposition tree structure. Moreover, we have to generate a minimal set $\mathcal{P}$ of axiom-links such that $(S^{\prec}, \mathcal{P})$ is a proof net. A question naturally arises: how could we directly compute such an appropriate set $\mathcal{P}$? This question is in fact related to the computation of a set of connections $\mathcal{SC}$ that linearly spans S, and vice-versa. In fact, we have the following result:

Theorem 4.3 *Let S be a sequent; a MLL connection net for S is a MLL proof net of S and vice-versa.*

Proof 4.3 *We have to prove the following result: a set of connections $\mathcal{SC}$ linearly spans S if and only if it corresponds to a set of axioms-links leading to a proof net.*

Consequently, a connection method could be derived from an existing proof method that constructs proof nets and their corresponding axiom-links. We will now focus on this direction taking into account our results on proof nets construction [24].
Let us remark that the natural corollary of the previous theorem is that any further connection method proposed for MLL, as for instance the one of [35] will be an alternative method for automated proof net construction.

4.5 Another example

Let us illustrate the connection characterization we have just defined with a new example. Let us consider the sequent $S\,:\vdash C \otimes (D \otimes A), A^{\perp} \wp B^{\perp}, B \otimes (D^{\perp} \wp\, C^{\perp})$.

The first step consists in constructing the formula tree corresponding to the formula
$C \otimes (D \otimes A) \wp A^{\perp} \wp B^{\perp} \wp B \otimes (D^{\perp} \wp\, C^{\perp})$ associated with polarity 0.
It is represented by the array of figure 3 including the main information.

The second step consists in constructing the set of the atomic paths that include the following elements:

(1) $\{a_2, a_8, a_9, a_{11}\}$ with the connection (a_9, a_{11})
(2) $\{a_2, a_8, a_9, a_{13}, a_{14}\}$ with the connection (a_2, a_{14})
(3) $\{a_4, a_8, a_9, a_{11}\}$ with the connection (a_9, a_{11})
(4) $\{a_4, a_8, a_9, a_{13}, a_{14}\}$ with the connection (a_4, a_{13})
(5) $\{a_5, a_8, a_9, a_{11}\}$ with the connections $(a_5, a_8), (a_9, a_{11})$
(6) $\{a_5, a_8, a_9, a_{13}, a_{14}\}$ with the connection (a_5, a_8)

Finally, from this set of connections, we can propose a subset $\mathcal{SC}$ defined by
$\mathcal{SC} = \{(a_2, a_{14}), (a_4, a_{13}), (a_5, a_8), (a_9, a_{11})\}$ such that $(\mathcal{FT}, \mathcal{SC})$ is a MLL connection net. From the previous results, we can claim that the corresponding net is in fact a MLL proof net.

u	pol(u)	lab(u)	Ptype(u)	Stype(u)
a_0	0	$C \otimes (D \otimes A) \wp (A^\perp \wp B^\perp) \wp (B \otimes (D^\perp \wp C^\perp))$	α	-
a_1	0	$C \otimes (D \otimes A)$	β	α_1
a_2	0	C	-	β_1
a_3	0	$D \otimes A$	β	β_2
a_4	0	D	-	β_1
a_5	0	A	-	β_2
a_6	0	$(A^\perp \wp B^\perp) \wp (B \otimes (D^\perp \wp C^\perp))$	α	α_2
a_7	0	$A^\perp \wp B^\perp$	α	α_1
a_8	0	$A^\perp$	α	α_1
a'_8	1	A	–	α_1
a_9	0	$B^\perp$	α	α_1
a'_9	1	B	–	α_1
a_{10}	0	$B \otimes (D^\perp \wp C^\perp)$	β	α_2
a_{11}	0	B	-	β_1
a_{12}	0	$D^\perp \wp C^\perp$	α	β_2
a_{13}	0	$D^\perp$	α	α_1
a'_{13}	1	D	-	α_1
a_{14}	0	$C^\perp$	α	α_1
a'_{14}	1	C	-	α_1

Figure 3: The formula tree of $\vdash C \otimes (D \otimes A), A^\perp \wp B^\perp, B \otimes (D^\perp \wp C^\perp)$.

5 Connection method and proof nets construction

In this section we illustrate the previous result about the relationships between a connection method and a procedure for proof nets construction in MLL. Therefore, we start to recall the principles of proof nets construction presented some years ago in [24] and we present a revised algorithm developed with B. Martin in [20]. Any algorithm dedicated to proof net construction could be used as a connection method but the one we consider combines the *connection calculus*, the *sequent calculus* and the *proof net calculus*. It means in fact that during the proof search for a given sequent S we can simultaneously construct the set of connections that linearly spans S, a sequent calculus proof of S and also a proof net of S, if it is provable.

The basic principles have been used in fact to develop new algorithms for MALL proof nets [20, 21] or for non-commutative proof nets [22]. The connection method based on it has an important specificity, compared to other approaches, is that it naturally avoids the sequent reconstruction from a proof net [6] or from the connection characterization (i.e. the subset of connections leading to provability) [44].

5.1 Proof nets construction

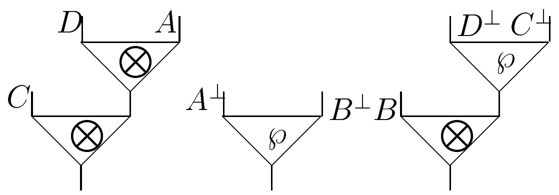
We recall here the principles developed in [24]. If we analyse the inductive definition of proof nets (see appendix A) we observe that starting from disjoint proof nets, we can construct new proof nets by fusion and extension with a $\otimes$-link from initial conclusions (considered as link premises). New proof nets can also be constructed, from a unique proof net, by extension with a $\wp$-link from initial conclusions (considered as link premises).

Then a natural construction can be based on the following idea: first we construct the decomposition tree of the given sequent, that will be used as a frame to guide the proof net construction. Moreover we consider a set $\mathcal{R}$ of intermediate proof nets that is initialized to the empty set. Then, starting from the leaves of the decomposition tree, we construct step by step axiom-links, that are elementary proof nets, added to the set $\mathcal{R}$. When conclusions of proof nets of $\mathcal{R}$ are in fact premises of a $\otimes$-link, we merge and extend them with this link to obtain a new proof net that replaces the two initial ones in $\mathcal{R}$. When conclusions of proof nets of $\mathcal{R}$ are in fact premises of a $\wp$-link, if they belong to the same proof net then we can extend it with this link to obtain a new proof net that replaces the initial one in $\mathcal{R}$ else we postpone the treatment of these conclusions. This construction of intermediate proof nets (subnets of the final proof net), that is essentially the role of the so-called `Propagation_construction` procedure, leads to the proof net if the initial sequent is provable or else to a failure. This propagation process is essential here to cover the semantics of the connectives during the construction process and is based on the inductive definition of appendix A.

As illustrated by the next example, it appears important to determine, if possible, some orders in the treatment of formulae that are conclusions of subnets and also premises of links. Moreover, two principles considered in this algorithm seem essential for this construction: the *inductive construction* of the subnets (elements of $\mathcal{R}$) that allows to avoid the exploration of useless branches and the *fusion principle* that means that a priority is to merge disjoint subnets with a$\otimes$-link as soon as possible. Considering the permutability results in linear sequent calculus, we know that we have, during a bottom-up proof-search, to apply a $\wp$ rule before a $\otimes$ rule and in the opposite order for a top-down strategy proof-search (from axioms to final sequent) [25]. The above inductive construction corresponds in fact to a top-down proof-search method, starting from axioms, here the axioms-links, and guided by the decomposition tree.

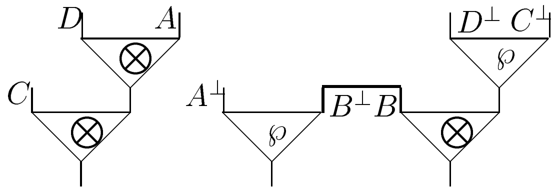
An example. To illustrate these construction principles, let us consider the sequent $S\ :\vdash C \otimes (D \otimes A), A^{\perp} \wp B^{\perp}, B \otimes (D^{\perp} \wp\ C^{\perp})$.

The decomposition tree Θ of S is the following figure



We do not include in this representation the initial $\wp$-links (linking the different branches) but should we want to cover the notion of formula tree of the connection method, we would have to do it. From this decomposition tree, we will construct some new structures (subnets) represented in bold on the figures.

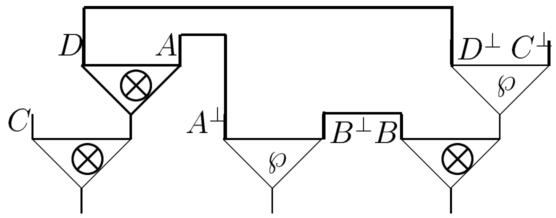
The choice of a first branch, on which we start the procedure, has to be arbitrary because no information on the structure is known. Let us assume that the algorithm starts with the branch $B \otimes (D^{\perp} \wp\ C^{\perp})$ and with the free leaf B. At first we treat B and $B^{\perp}$ and construct the subnet $(B, B^{\perp})$ (in bold in the figure).



At this stage, we could try to deal with $A^{\perp}$ (of the $A^{\perp} \wp\ B^{\perp}$ branch) to have two elementary proof nets with conclusions being premises of a link and then try to apply propagation. But in

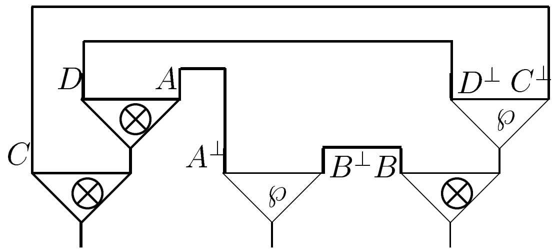
this case, it is a $\wp$-link and then the extension is impossible.
Then the point is to start from the other premise of the $\otimes$-link, namely $D^\perp \wp\ C^\perp$, in such a way to treat this $\otimes$-link as soon as possible. In fact, it is a $\wp$-link and we could prefer to treat another $\otimes$-link but we forbide to treat a leaf of a new branch because we want to apply the fusion principle as soon as possible. As none of these premises $(D^\perp, C^\perp)$ is treated, the treatment of one of them is possible without problem (here we consider $D^\perp$). If there were only $\wp$-links such that one of the premises was treated in $\mathcal{R}$, then elements of $\mathcal{R}$ could not merge and thus this attempt would be unsuccessful.

After connecting $D^\perp$ and D (of $D \otimes A$), as we try to deal at first with $\otimes$-links, we have to consider the leaf A to treat the $D \otimes A$ link. After having built the axiom-link $(A, A^\perp)$ we have both conclusions D and A of two elementary proof nets that can be merged and extended with a $\otimes$-link into a new subnet.



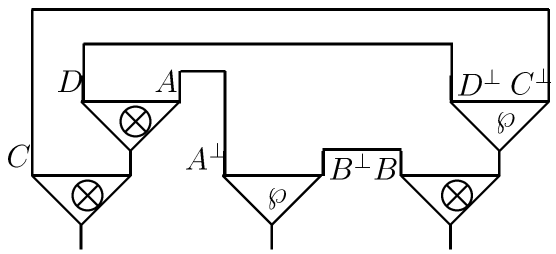
As a consequence of the construction of the $(A, A^\perp)$ axiom-link, both premises of the $A^\perp \wp\ B^\perp$ branch that are treated (are conclusions of some substructures) and then the propagation-construction can be possibly applied. But as these premises belong to two different subnets, such extension is forbidden and then the link $A^\perp \wp B^\perp$ will be now considered as *in waiting position.*

The link $D \otimes A$ being treated, we will now consider the leaf C and the propagation construction on the subnets is now successful.



From this step, we can apply the propagation to the $D^\perp \wp\ C^\perp$ link because both premises belong to the same subnet. Then, as this second premise of the branch we start with is treated we can apply the propagation from B and this one and obtain the following subnet

The $\wp$-link that was in a waiting position $(A^\perp \wp\ B^\perp)$ has its both premises belonging to the same subnet and then the propagation is now possible to obtain the final proof net, that covers the initial decomposition tree.



The procedure then considers in priority the branches that are above a $\otimes$-link compared to those above a $\wp$-link. This choice is not primordial for the success of the algorithm but simply corresponds to an heuristic that consists in postponing the arbitrary choices as late as possible to avoid them if possible. In fact, this idea of $\wp$-link in waiting position and its use inside the structure construction process with the induced order characterize the revised version of the algorithm developed with B. Martin [20, 21]. In fact, its basic principles have been mainly applied in new developments for proof nets in non-commutative MLL [22] and in extension with the additives [20, 21].

The procedure (cf. [20]). The algorithm, following the presentation of [20], with the principal procedure named MLL_PN, deals with three main data-structures: a list $\wp$`_waiting` of $\wp$-links untreated but with 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 sub-formulas in the structure under construction, the branches of which are to be treated.
It is composed of the following four procedures: the initial call of the main one, the MLL_PN procedure, on the input sequent we have to prove, will return either a failure message or a proof structure (that is in fact a 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.

Procedure MLL_PN ($S:$ MLL sequent) $\rightarrow$ `failure` or $\Pi:$ proof structure of S

1. construct Θ, the decomposition tree of S;
2. verify the duality property[2], else return(`failure`);
3. initialize the lists `f_to_treat`, $\wp$`_to_treat` and $\wp$`_waiting` to $\emptyset$;
   choose a conclusion of S; add it to `f_to_treat`;
4. initialize $\mathcal{R} := \emptyset$;
5. $\mathcal{R}$:= `Search_free_leaf_to_treat(`$\mathcal{R}$, `f_to_treat`, $\wp$`_to_treat`, $\wp$`_waiting)`;
6. Let Π be the unique element of $\mathcal{R}$, Return(Π).

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

1. While one of the three lists is not empty, do
   (a) if `f_to_treat` is not empty then consider its first element A and remove it from the list;
      - if A is a free leaf then
        `Treat_free_leaf(A,` $\mathcal{R}$`, f_to_treat,` $\wp$`_to_treat,` $\wp$`_waiting)`;
      - if A has $\otimes$ as principal connective then its both premises are placed at the beginning of `f_to_treat`;
      - if A has $\wp$ as principal connective then if its both premises were not treated then A is added to $\wp$`_to_treat`;

[2] Duality property: if a sequent is provable in MLL then the multi-set of its atomic formulae can be splitted into pairs of dual formulae.

(b) else if $\wp$_to_treat is not empty then consider its first element; remove it from the list; if the two premises of this $\wp$-link are not already treated then add one of them in f_to_treat;

(c) else for each link of $\wp$_waiting (that is not empty) and A one of its premises do
$\mathcal{R}'$:= Propagation_construction(A, true, R, f_to_treat, $\wp$_waiting);
if all links lead to a failure then return(failure) else return($\mathcal{R}'$)[3];

2. if all leaves are treated then return($\mathcal{R}$) else return(failure).

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

1. choose a free leaf B dual to A

   (a) create an axiom-link between A and B; add it to $\mathcal{R}$.

   (b) $\mathcal{R}'$:= Propagation_construction(A, false, $\mathcal{R}$, f_to_treat, $\wp$_waiting);

   (c) if ($\mathcal{R}' \neq$ failure) then
   $\mathcal{R}'$:= Propagation_construction(B, true, $\mathcal{R}'$, f_to_treat, $\wp$_waiting);

   (d) if ($\mathcal{R}' \neq$ failure) then
   $\mathcal{R}'$:= Search_free_leaf_to_treat($\mathcal{R}'$, f_to_treat, $\wp$_to_treat, $\wp$_waiting);

2. $\mathcal{R}$:= $\mathcal{R}'$.

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

1. While B (the other premise of the link having the current subformula A) is already treated, do

   (a) if it is a $\otimes$-link (for instance A$\otimes$B) then
      - if its premises belong to two disjoint elements of $\mathcal{R}$, then merge and extend them with a $\otimes$-link and add this structure in $\mathcal{R}$ from where the two initial 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 this element is extended with the link to give a new element of $\mathcal{R}$ from 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.

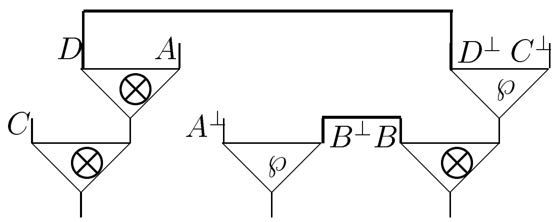
2. if (prop = true) and the link under the current sub-formula is a $\otimes$-link then its other premise is added at the end of f_to_treat.

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

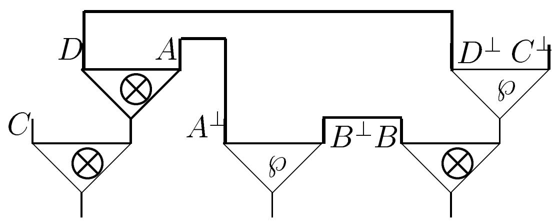
In summary, the procedure chooses axiom-links and uses the initial semantical information (about connectives) that are implicitly involved in the decomposition tree to construct, if possible, step by step (and from the atomic formulae) a structure that will be a proof structure and even a proof net. The central point is the Propagation_construction that intuitively justifies that the set of connections we construct step by step could potentially span the initial formula.

[3] Let us remark that $\mathcal{R}'$ can be a failure in this case.

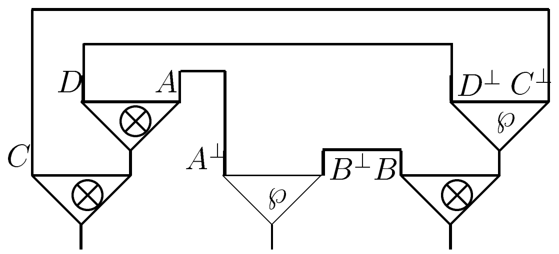
The example. Let us run this procedure on the previous example used to illustrate the main principles of the construction.
After the construction of the decomposition tree, the subformula $B \otimes (D^{\perp} \wp\, C^{\perp})$ is added to `f_to_treat`. The `Search_free_leaf_to_treat` procedure is called with $\mathcal{R}$ being empty. The list `f_to_treat` being not empty we consider its first element $B \otimes (D^{\perp} \wp\, C^{\perp})$ and `f_to_treat` becomes empty. The principal connective is $\otimes$ and thus the subformulae B and $D^{\perp} \wp\, C^{\perp}$ are added to `f_to_treat`. As it is not empty, we consider its first element B that is suppressed from it and call `Treat_free_leaf` on B. There is only one $B^{\perp}$, the corresponding axiom-link $(B, B^{\perp})$ is added to $\mathcal{R}$. The two calls of `Propagation_construction` do not modify $\mathcal{R}$ and the three main lists because $B^{\perp}$ is a premise of a $\wp$-link. Then `Search_free_leaf_to_treat` is called and the element $D^{\perp} \wp\, C^{\perp}$ of `f_to_treat` is added in $\wp$`_to_treat` that is not empty now. Then one of the premises of the $D^{\perp} \wp\, C^{\perp}$ link, namely $D^{\perp}$, is included in `f_to_treat`. By `Treat_free_leaf` the axiom-link $(D, D^{\perp})$ is added to $\mathcal{R}$ and the two calls of `Propagation_construction` do not change $\mathcal{R}$ and add A to `f_to_treat`.
At this point we have the following situation:



Then, the free leaf A is treated and the corresponding axiom-link is added to $\mathcal{R}$. The first call of `Propagation_construction` merges the axiom-links $(A, A^{\perp})$ and $(D, D^{\perp})$ by extension with the $D \otimes A$ link. Then $\mathcal{R}$ includes this new element and $(B, B^{\perp})$. Moreover C is added to `f_to_treat`. The second call includes $A^{\perp} \wp B^{\perp}$ in $\wp$`_waiting` because its premises do not belong to the same element of $\mathcal{R}$.

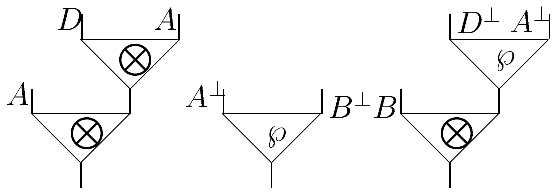


Then we treat C in a similar way adding the axiom-link $(B, B^{\perp})$ to $\mathcal{R}$. The calls of `Propagation_construction` lead to the merge of all the elements of $\mathcal{R}$ into only one element.



Then the link $A^{\perp} \wp B^{\perp}$ in the $\wp$`_waiting` is considered and the last `Propagation_construction` leads to the final net because the premises belong to the same element of $\mathcal{R}$.

Remarks. Let us remark at first that we add a $\wp$-link to `℘_waiting` either, as in this example, because another $\wp$-link, the treatment of which will lead to the fusion of the elements of $\mathcal{R}$ including its premises, is not yet treated, or because a $\otimes$-link with only one premise treated does not link at this point the premise to its conclusion.
It is important to notice that in our example there is only one occurrence of each atomic formula and thus the choice of a dual free leaf is immediate. If there is more then one occurrence of an atomic formula then we make an arbitrary choice (see Procedure `Treat_free_leaf`). We can modify this procedure including a loop to deal with "all the dual free leaves of A". It means that the previous treatment of the free leaves will be done inductively for all the possibilities of axiom-links creation. In this case, the modified procedure MLL_PN has no more a proof net of a given sequent S as output but the set of all the proof nets of S. For instance, if we consider our example where C is replaced by A (and then $C^{\perp}$ by $A^{\perp}$) then we have the sequent $S\,:\vdash A\otimes(D\otimes A), A^{\perp}\wp B^{\perp}, B\otimes(D^{\perp}\wp\, A^{\perp})$ with the following decomposition tree



During the procedure execution, we can make the same choice for the axiom-link $(A, A^{\perp})$ and then have the same steps that lead to the similar proof net. If we make the alternative choice for $(A, A^{\perp})$ the rest of execution is different with early successful propagation-constructions and we obtain a different proof net for S.

The properties. After the description of the procedure we can analyse its properties. In fact the resulting proof structure is necessarily a proof net. We then have a correct and complete algorithm for proof nets construction in MLL [24, 20].

Theorem 5.1 (correctness) *If the algorithm returns a proof structure corresponding to the sequent S, then S is provable in MLL.*

Proof 5.1 *Let us prove that the proof structure obtained by the algorithm is a proof net.*
It is sufficient to prove that the elements of $\mathcal{R}$ are proof nets, at any moment. It is direct because they are constructed using the inductive definition of proof nets.
Moreover, it is sufficient to verify that this proof net is a proof net of S. It is direct because the net is built from the pre-proof structure (or decomposition tree) associated to S.

Theorem 5.2 (completeness) *If S is a sequent provable in MLL then the algorithm returns a proof net of S.*

Proof 5.2 *Let us assume that it is false, i.e., during its execution, this algorithm returns a failure message but there exists a proof net of S having its axiom-links already built when it fails. As any MLL proof net can be constructed in an inductive way, we can thus continue the inductive construction from the set $\mathcal{R}$.*
Let L be a $\otimes$-link or a $\wp$ link that corresponds to the first rule applied during this construction, a premise of which being a conclusion of an element of $\mathcal{R}$;

- *if L is a $\otimes$-link, then, during the treatment of its first premise, the second premise is added to* `f_to_treat`*. Thus, some terminal branches of the second premise were treated and then the second premise was necessarily treated during the failure.*
  In fact, if it is not the case, L would not be the first. But in the rest of the treatment of the second premise, the link L not treated involves a failure case. Then its premises belong to the same element of $\mathcal{R}$; which is contradictory;

- *if L is a $\wp$-link then, since the failure of the procedure, the inductive construction is only able to add isolated axiom-links possibly connected together. Nothing is added to the elements of $\mathcal{R}$. Thus, its two premises have been already treated but the case of this link, that was necessarily refused, has been studied just before the failure; which is contradictory.*

The algorithm necessarily terminates on a provable sequent and then returns a proof net of S by the correction theorem.

This procedure have been studied [24, 20] as a preliminary step before considering proof nets construction in other linear logic fragments as non-commutative multiplicative linear logic [22] or additive and multiplicative linear logic [20, 21].

5.2 A connection method algorithm

For the connection method the final interesting result is the set of connections involved in the construction of the final structure. We could thus modify the output of the procedure to obtain this set of connections. As the final structure is a proof net, the connections linearly span the initial sequent and imply its provability.

Theorem 5.3 (correctness) *If the procedure returns a final set of connections then the input sequent S is provable in MLL.*

Proof 5.3 *Let $\mathcal{P}$ be the set of connections obtained as a result of the procedure. We can prove that $\mathcal{P}$ is a minimal path that satisfies the conditions (α) and (β) of theorem 2.1. The success of the application of the propagation procedure implies that these conditions are satisfied.*

Theorem 5.4 (completeness) *If S is a sequent provable in MLL then the procedure returns a set of connections that span the sequent.*

Proof 5.4 *The proof is given using an indirect argument that is the completeness of the procedure for the proof nets construction (see previous subsection).*

The example (continued). If we consider the positions of the formula corresponding to the decomposition tree and if we run the procedure on this tree we obtain the following set as output $\mathcal{SC} = \{(a_2, a_{14}), (a_4, a_{13}), (a_5, a_8), (a_9, a_{11})\}$. If we trace the procedure we observe that it finds at first (a_4, a_{13}) then (a_5, a_8) and successively (a_9, a_{11}) and (a_2, a_{14}) to finally return this set $\mathcal{SC}$.

The main result is that a procedure for MLL proof net construction can be directly considered as a connection method for this fragment. It is illustrated here with one given algorithm that moreover has interesting characteristics w.r.t. proof construction. We could modify (or specialize) the previous procedure to directly present a connection method but the interesting result is in fact the relationships between proof nets deduction and such a method, as illustrated by this algorithm. Therefore, we could substitute another procedure for proof nets construction

and then obtain another based-on connection proof-search method.
This analysis and the underlying ideas about these connection characterization and method for MLL have been presented at first in 1996 [17] after being conjectured some years before. Recently and independently, another work on proof-search in linear logic based on the connection method [35] confirms the interest of such a method to prove linear logic sequents. A dual result is that it could be used as an algorithm for proof nets construction in MLL. Comparisons from the both points of view (connections and proof nets) could be developed in future work taking into account algorithmic and implementation aspects.

5.3 A top-down proof-search algorithm

The proof-search method we have presented for MLL combines in fact the *connection calculus*, the *proof net calculus* but also the *sequent calculus*. It means that during the proof-search we simultaneously construct not only a set of connections that span the sequent and a proof net but also a sequent proof. In fact, this method can be used to directly construct a sequent proof from a MLL sequent. It constructs step by step the corresponding sequent proof following a top-down (from axioms to formula to prove) approach. More precisely, from 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 $\otimes$ or $\wp$ inference rules. Moreover, what we have called substructures (or subnets) correspond here to subproofs of the final proof.

Let us illustrate this point with the sequent $S : \vdash C \otimes (D \otimes A), A^{\perp} \wp B^{\perp}, B \otimes (D^{\perp} \wp\, C^{\perp})$.
The first link (or connection) $(B, B^{\perp})$ corresponds to the axiom $\vdash B, B^{\perp}$ and then we construct the axioms $\vdash D, D^{\perp}$ and $\vdash A, A^{\perp}$. The application of `Propagation-construction` postpones the use of $\vdash B, B^{\perp}$ and applies the $\otimes$ inference rule to the others to obtain the sequent $\vdash D \otimes A, D^{\perp}, A^{\perp}$. Then the procedure chooses $\vdash C, C^{\perp}$ as the next possible axiom and `Propagation-construction` applied to both sequents returns at first $\vdash C \otimes (D \otimes A), A^{\perp}, D^{\perp}, C^{\perp}$ and then $\vdash C \otimes (D \otimes A), A^{\perp}, D^{\perp} \wp\, C^{\perp}$. The procedure now considers the postponed axiom $\vdash B, B^{\perp}$ to construct at first the sequent $\vdash C \otimes (D \otimes A), A^{\perp}, B^{\perp}, B \otimes (D^{\perp} \wp\, C^{\perp})$ and finally $\vdash C \otimes (D \otimes A), A^{\perp} \wp B^{\perp}, B \otimes (D^{\perp} \wp\, C^{\perp})$. Thus the procedure returns the following sequent proof:

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

This top-down proof-search method can be seen as a forward reasoning method but more goal-oriented as the usual ones. In fact, it starts from axioms and generates new formulae (or sequents) until the goal formula (or sequent) is reached and goal-oriented by the use of the decomposition tree of this goal. Hence it corresponds to the results about top-down proof-search based on permutability properties with the construction of canonical proofs [25]. Moreover, we have criteria to only select some relevant axioms and inference rules which are used to develop the sequent proof. These underlying ideas are in fact the foundations of the Maslov's inverse method [36] and it seems, modulo some correspondences between concepts, that our top-down

proof-search method can be considered as an inverse method for MLL. This important and interesting property is confirmed by works on the inverse method and its implementation [12] where concepts like "assemblages" or "gluing of assemblages" are used. A deeper analysis of these relationships will be done in further work.
The efficiency of automated proof methods could strongly depend on a compact representation of a proof. The characterization of validity, on which connection methods are based, avoids some redundancies contained for instance in sequent calculus. But the problem is that such an automatically generated proof (the set of connections that span the sequent) is often difficult to understand. This point explains some attempts about the conversion of machine proofs into a humanly comprehensible form and for instance into sequent calculus proofs that can be considered as more human-oriented than matrix proof, and thus can be used within some interactive proof environments. In this context, some works have considered the sequent reconstruction problem (from a set of connections) and then have developed transformations from matrix proofs into sequent proofs for intuitionistic logic [45]. Our connection method avoids this problem of sequent reconstruction because of its ability to generate directly a sequent proof. Moreover as it is initially a method for proof nets construction, it is clear than when a proof net is constructed then a corresponding sequent proof can be simultaneously constructed. It is well-known that mapping sequent proofs to proof nets is almost trivial while showing the converse (from proof nets to proofs) requires a lot of work. The corresponding sequent reconstruction (from a proof net) that has been previously solved by different techniques, as for instance the sweepline technique [6] is not necessary with our approach.

6 Dealing with the additives

A natural question arises from the previous results: can we extend our approach to the additives connectives ($\oplus, \&$) and then to logical fragments including them, as for instance the Multiplicative and Additive Linear Logic (MALL) ? In fact there exists some works on automated proof-search in the MALL fragment but even if we are able to reduce some sources of non-determinism during the proof-search process, it is not so evident to have a good management of the additives [25, 38]. We have recently defined an algorithm for the proof nets construction for MALL [20, 21] that could be used for the ALL fragment but as far as we know no connection method has been proposed for these fragments. For the proof-search in MALL, we have also considered an alternate approach based on the idea to only study the Additive Linear Logic (ALL) [19] and its proof-theoretical properties and then to see MALL as its extension with the multiplicatives. In a similar way, we aim at considering at first the apparently simple fragment ALL to address the question of the definition of a connection characterization and a method in a fragment including additives.

6.1 Proof-search in ALL

We now consider the additive fragment of LL without constants (ALL) where the only connectives in formulae and sequents are the additive ones, i.e. $\&$ (conjunction) and $\oplus$ (disjunction), and the linear negation $^{\perp}$. In general, these connectives are interesting for different reasons but involve some problems due to the context dependency [28]. We have already considered this fragment, that is often included in other LL fragments used in programming applications and some proof-search methods for ALL have been studied to obtain for instance procedures in polynomial time [18, 19].
In this section we recall definitions and proof theoretic properties of ALL already studied in [18, 39], starting from the initial sequent calculus for this fragment [19].

Sequents in additive linear logic (ALL) are of the following form $E \vdash F$, where E and F are formulae, since we have neither multiplicative connectives nor structural rules. The inference system for ALL is the following:

$$\frac{}{E \vdash E}\ Ax \qquad \frac{E_1 \vdash F \quad F \vdash E_2}{E_1 \vdash E_2}\ cut$$

$$\frac{E_1 \vdash F}{E_1 \& E_2 \vdash F}\ \&_{L_1} \quad \frac{E_2 \vdash F}{E_1 \& E_2 \vdash F}\ \&_{L_2} \quad \frac{E \vdash F_1 \quad E \vdash F_2}{E \vdash F_1 \& F_2}\ \&_R$$

$$\frac{E_1 \vdash F \quad E_2 \vdash F}{E_1 \oplus E_2 \vdash F}\ \oplus_L \quad \frac{E \vdash F_1}{E \vdash F_1 \oplus F_2}\ \oplus_{R_1} \quad \frac{E \vdash F_2}{E \vdash F_1 \oplus F_2}\ \oplus_{R_2}$$

Figure 4: A one-conclusion ALL proof system

Moreover we have a negation, denoted $^\perp$,which is involutive, $A^{\perp\perp} = A$, and which verifies De Morgan laws: $(A \& B)^\perp = A^\perp \oplus B^\perp$ and $(A \oplus B)^\perp = A^\perp \& B^\perp$. Here we restrict ourselves, without loss of generality, to sentences where negation is applied only to atoms. Considering the proof-search we remark that there are non-deterministic choices coming from the additive connectives and we would like to delay this choice and try to accumulate a great deal of information before exploring branches of the proof tree.
In fact, we defined another sequent calculus with multi-sets of formulas as antecedents and succedents (as for the MLL fragment) more adapted for automated proof-search. That is why we have defined the notion of additive sequent and then a corresponding sequent calculus for ALL. This system is different in the following respect with a deductibility relation $\vdash$ on additive contexts that are defined as follows:
Let $\Gamma = \{A_1, \ldots, A_n\}$ and $\Delta = \{B_1, \ldots, B_m\}$ be two contexts which are multi-sets of formulae, an additive sequent has the form $A_1 \colon \ldots \colon A_n \vdash B_1 \colon \ldots \colon B_m$. Intuitively, following the classical interpretation, $A_1 \colon \ldots \colon A_n \vdash B_1 \colon \ldots \colon B_m$ means that the additive conjunction $A_1 \& \ldots \& A_n$, entails the additive disjunction $B_1 \oplus \ldots \oplus B_n$. But what is the meaning of this entailment ? In fact $A_1 \colon \ldots \colon A_n \vdash B_1 \colon \ldots \colon B_m$ holds if there exists some $A_i \in \Gamma$ and some $B_j \in \Delta$ such that $\& A_i \vdash \oplus B_j$. Thus in an additive sequent, the symbol :, that replaces the comma, stands for & on the left-hand side and for $\oplus$ on the right-hand side.
Let us consider now the additive sequent calculus, denoted ALL_{mc} , for the additive fragment of linear logic, where sequents have multi-conclusions [18, 39]. As it has the cut-elimination property we can expect to use it as a starting system to define a based-on-connection characterization for this fragment.

It is important to notice the particularities of the $\&_R$ and $\oplus_L$ rules which are crucial during proof-search. During a bottom-up proof construction if we apply $\&_R$ rule, the formula $F_1 \& F_2$ is chosen among the succedent (it is the principal formula) and the proof is divided into two parts, following the usual pattern. But the other succedent formulae, that are included in the set Δ, are erased. Such rules are similar to the so-called special rules of multi-conclusion sequent calculus for intuitionistic logic [43, 48].
Assuming that we would like to deal with negations instead of just considering formulae through De Morgan laws, we should also add the following rules:

$$\frac{\vdash A, \Delta}{\Gamma \colon A^\perp \vdash \Delta}\ \perp_L \qquad \frac{\Gamma \vdash \Delta \colon A^\perp}{\Gamma \colon A \vdash}\ \perp_R$$

Let us consider $G \& (E \oplus (A \& B)) \vdash ((A \oplus C) \& (B \oplus D)) \oplus E \oplus F$ to illustrate the use of this

$$\frac{}{\Gamma{:}A \vdash A{:}\Delta}\;Ax$$

$$\frac{\Gamma_1 \vdash \Delta_1{:}E \quad E \vdash \Delta_2}{\Gamma_1{:}\Gamma_2 \vdash \Delta_1{:}\Delta_2}\;cut_1 \qquad \frac{\Gamma_1 \vdash E \quad E{:}\Gamma_2 \vdash \Delta_2}{\Gamma_1{:}\Gamma_2 \vdash \Delta_1{:}\Delta_2}\;cut2$$

$$\frac{\Gamma{:}E_1{:}E_2 \vdash \Delta}{\Gamma{:}E_1\&E_2 \vdash \Delta}\;\&_L \qquad \frac{\Gamma \vdash F_1 \quad \Gamma \vdash F_2}{\Gamma \vdash F_1\&F_2{:}\Delta}\;\&_R$$

$$\frac{E_1 \vdash \Delta \quad E_2 \vdash \Delta}{\Gamma{:}E_1 \oplus E_2 \vdash \Delta}\;\oplus_L \qquad \frac{\Gamma \vdash F_1 \oplus F_2{:}\Delta}{\Gamma \vdash F_1{:}F_2{:}\Delta}\;\oplus_R$$

Figure 5: A multi-conclusion ALL proof system

proof system. In fact, we can build the following proof

$$\frac{\dfrac{\dfrac{\dfrac{}{E \vdash (A \oplus C)\&(B \oplus D){:}E{:}F}\;Ax \quad \dfrac{\dfrac{\dfrac{\dfrac{}{A{:}B \vdash A{:}C}\;Ax}{A{:}B \vdash A \oplus C}\;\oplus_R \quad \dfrac{\dfrac{}{A{:}B \vdash B{:}D}\;Ax}{A{:}B \vdash B \oplus D}\;\oplus_R}{A{:}B \vdash (A \oplus C)\&(B \oplus D){:}E{:}F}\;\&_R}{A\&B \vdash (A \oplus C)\&(B \oplus D){:}E{:}F}\;\&_L}{G{:}(E \oplus (A\&B)) \vdash (A \oplus C)\&(B \oplus D){:}E{:}F}\;\oplus_L}{G{:}(E \oplus (A\&B)) \vdash ((A \oplus C)\&(B \oplus D)) \oplus E \oplus F}\;\oplus_R}{G\&(E \oplus (A\&B)) \vdash ((A \oplus C)\&(B \oplus D)) \oplus E \oplus F}\;\&_L$$

The same example developed in the initial ALL system leads to non-determinisms for the application of the $\&_L$ and $\oplus_R$ rules. Let $\Gamma \vdash \Delta$ be an additive sequent and $\Gamma = \{E_1{:}E_2{:}\ldots{:}E_n\}$ and $\Delta = \{F_1{:}F_2{:}\ldots{:}F_m\}$ be two contexts, we define the two sets $\&$ and $\oplus$ as follows: $\&\Gamma =_{def} \Gamma\{E_{i_1}\&\ldots\&E_{i_k}/E_{i_j} \in \Gamma\}$ and $\oplus\Delta =_{def} \{F_{i_1} \oplus \ldots \oplus F_{i_l}/F_{i_p} \in \Delta\}$.

Proposition 6.1 *(i) Let E and F be two formulae, if $E \vdash F$ holds in ALL then $E \vdash F$ holds in ALL_{mc} . (ii) Let Δ and Γ be two contexts, if $\Gamma \vdash \Delta$ holds in ALL_{mc} then there exists two formulae $E \in \&\Gamma$ and $F \in \oplus\Delta$ such that $E \vdash F$ holds in ALL.*

Theorem 6.1 *Let E and F be two formulae, $E \vdash F$ holds in ALL if and only if $E \vdash F$ holds in ALL_{mc} .*

Proof 6.1 *Immediate result from proposition 6.1.*

Moreover the system ALL_{mc} verifies the cut elimination property [18, 39]. Indeed, this property is important to propose efficient proof-search procedures because it implies the subformulae property. We can also mention that proof-search can be based on possible proof transformations justified by permutability results. Then, having recalled some aspects of proof-search in ALL, we can study a possible connection characterization from the latter system.

6.2 A Connection characterization of ALL

An initial sequent $A_1, ..., A_n \vdash B_1, .., B_m$ corresponds to $A_1\&..\&A_n \vdash B_1 \oplus .. \oplus B_m$ and is in fact similar to a MLL sequent where $\&$ and $\oplus$ respectively replace $\otimes$ and $\wp$. Then a similar connection characterization could be expected with some similar notions as those of path and connection. As in the multi-conclusion sequent calculus for intuitionistic logic [48], we have rules (called *special rules*) where some set of formulas are erased when these logical rules are applied. The validity will be characterized in term of spanning set of complementary connections, the notion of *complementarity* taking into account some non-permutability results. For instance, if we consider the previous example of proof we observe that, in a bottom-up proof-search, the rule $\oplus_L$ has to be applied before the $\&_R$. The *special* rules, that are here $\&_R$, $\perp_R$ and $\oplus_L$, introduce this problem into the sequent search space. As we have to ensure that formulae occurrences containing (as subformulae) the two atomic formulae of a connection are not deleted by the application of a special rule, we have to control the dual formulae of the special rules. But let us present the main definitions.

Definition 6.1 *A* signed formula *is a pair $< A, n >$ where A is a formula and $n \in \{0, 1\}$. It is said* atomic *if its constituent formula is atomic.*
A signed formula of the form $< A\&B, 1 >$, $< A \oplus B, 0 >$, $< A^{\perp}, 1 >$ or $< A^{\perp}, 0 >$ is of conjunctive type (or α-type). We define the components α_1, α_2 as follows:

α	α_1	α_2
$< A\&B, 1 >$	$< A, 1 >$	$< B, 1 >$
$< A \oplus B, 0 >$	$< A, 0 >$	$< B, 0 >$
$< A^{\perp}, 1 >$	$< A, 0 >$	$< A, 0 >$
$< A^{\perp}, 0 >$	$< A, 1 >$	$< A, 1 >$

A signed formula of the form $< A\&B, 0 >$ or $< A \oplus B, 1 >$ is of disjunctive type (or β-type). We define the components β_1, β_2 as follows:

β	β_1	β_2
$< A\&B, 0 >$	$< A, 0 >$	$< B, 0 >$
$< A \oplus B, 1 >$	$< A, 1 >$	$< B, 1 >$

Definition 6.2 *A signed formula is a* special formula *if it is atomic or it is a negation or if its major connective is either the* with *conjunction $\&_R$ or the* plus *disjunction $\oplus_L$.*

The notion of *formula tree, positions, labels* and *polarity* are defined roughly as before. But we have to define a special type *Spt* for formulae as follows:

Definition 6.3 *If the polarity of a formula $\mathcal{F}$ that is atomic or with $\&$, $\perp$ as principal connective is (resp. with $\oplus$) is 1 (resp. 0) then its special type is ϕ_0 ($\in \Phi_0$).*
If the polarity of a formula $\mathcal{F}$ that is atomic or with $\&$, $\perp$ as principal connective is (resp. with $\oplus$) is 0 (resp. 1) then its special type is ψ_0 ($\in \Psi_0$).

If we consider a signed formula $< F, 0 >$ we could add *extra positions* between each position that correspond to a special formula and its parent . Then the secondary type of this extra position will be either ϕ_0 or ψ_0 (see [48]). For simplification of the presentation we only use the special type.

An example. Let us illustrate this presentation with the following ALL sequent S_0, i.e., $G\&(E \oplus (A\&B)) \vdash ((A \oplus C)\&(B \oplus D)) \oplus E \oplus F$.
The first step consists in constructing the indexed formula tree (including labels, polarities denoted by 0 and 1 and the principal and secondary types) as follows.

The notions of *path through S* and of *connection* are defined as in the MLL case.

u	pol(u)	lab(u)	Ptype(u)	Stype(u)
a_0	-	S_0	-	-
a_1	1	$G\&(E \oplus (A\&B))$	α	-
a_2	1	G	-	α_1
a_3	1	$E \oplus (A\&B)$	β	α_2
a_4	1	E	-	β_1
a_5	1	$A\&B$	α	β_2
a_6	1	A	-	α_1
a_7	1	B	-	α_2
a_8	0	$((A \oplus C)\&(B \oplus D)) \oplus E \oplus F$	α	-
a_9	0	$((A \oplus C)\&(B \oplus D)) \oplus E$	α	α_1
a_{10}	0	$(A \oplus C)\&(B \oplus D)$	β	α_1
a_{11}	0	$A \oplus C$	α	β_1
a_{12}	0	A	-	α_1
a_{13}	0	C	-	α_2
a_{14}	0	$B \oplus D$	α	β_2
a_{15}	0	B	-	α_1
a_{16}	0	D	-	α_2
a_{17}	0	E	-	α_2
a_{18}	0	F	-	α_2

Figure 6: The formula tree of $G\&(E \oplus (A\&B)) \vdash ((A \oplus C)\&(B \oplus D)) \oplus E \oplus F$.

Definition 6.4 *Let S be a sequent, a* path through S *is a subset of the positions of its formula tree defined as follows:*
The set of paths through S is the smallest set such that:
1. $\{a_0\}$ is a path, where a_0 is the root position of the formula tree for F;
2. if s, α is a path then $(s \setminus \{\alpha\}), \alpha_1, \alpha_2$ is a path;
3. if s, β is a path then $(s \setminus \{\beta_1\})$ and $(s \setminus \{\beta_2\})$ are paths.

Definition 6.5 *An* atomic path *through S is a path only composed of atomic formulae.*

But here we have to refine the notion of connection into that of *complementary connection* using the notion of *prefix* of a position that encodes the context of that position within the formula tree with respect to the special positions [48]. The crucial issue consists in characterizing when a connection is complementary and thus corresponds to an instance of the initial sequent.

Definition 6.6 *Let T be the union of Ψ_0 and Φ_0, we associate to each position u of the formula tree a sequence of positions $pre(u) = u_1, u_2, .., u_n$, called* prefix *such that $u_1 \ll u_2 \ll ... \ll u_n \ll u, 1 \leq n$ are the elements of T that dominate u in the formula tree.*

The prefix of a position encodes the context of that position within the formula tree with respect to the special positions. The final step is to define complementarity for a connection in terms of the prefix of the atomic positions. It is similar to the treatment of provability in intuitionistic logic in [48].

Definition 6.7 *The accessibility relation on T^* is the smallest relation such that either $p = qu$ with $u \in T$ or $q = p$ or $p \prec q$.*
An all-substitution is a mapping σ from Ψ_0 to T^* (that induces a relation $\sqsubset$ on $T \times T$). It is said admissible *if σ respects admissibility and $(\ll \cup \sqsubset)^+$ is irreflexive.*

Definition 6.8 *A* complementary connection (u, v) *is a connection* $(lab(u) = lab(v))$ *with* $\sigma^\sharp(pre(u)) = \sigma^\sharp(pre(v))$, $\sigma^\sharp$ *being the homomorphic extension of* σ *to* T^*.

Definition 6.9 *A set* $\mathcal{SC}$ *of connections* spans *the sequent* S *if*
(a) all atoms in the formula tree of S *appear* at most once *with polarity* 0 *or* 1 *in* $\mathcal{SC}$*;*
(b) if (P, P') *and* (Q, Q') *distinct elements of* $\mathcal{SC}$ *then* $P \neq Q$ *and* $P' \neq Q'$*;*
(c) every atomic path contains a complementary connection from $\mathcal{SC}$*;*
(d) it is minimal (no proper subset of SC *spans* S*).*

Another way to present the previous results on connection characterization of the ALL fragment is the following:

Definition 6.10 *An* ALL connection structure *for the sequent* S *is a pair* $(\mathcal{FT}, \mathcal{SC})$ *where* $\mathcal{FT}$ *is the formula tree and* $\mathcal{SC}$ *is a subset of the set of all the connections such that all atoms in* S *appear in at most one connection of* $\mathcal{SC}$*.*
An ALL connection structure $(\mathcal{FT}, \mathcal{SC})$ *is* complementary *if each atomic path through* S *contains a complementary connection of* $\mathcal{SC}$*.*
An ALL connection structure $(\mathcal{FT}, \mathcal{SC})$ *is* minimal *if the set* $\mathcal{SC}$ *is minimal, i.e., if we drop a connection then the complementarity property is lost.*

Definition 6.11 *An* ALL connection net *for the sequent* S *is an ALL connection structure* $(\mathcal{FT}, \mathcal{SC})$ *that is complementary and minimal.*

Theorem 6.2 *An ALL sequent* S *is provable if and only if there exists an ALL connection net for* S*.*

Let us conclude this section with the characterization of the existence of a sequent proof of a formula and hence of its validity in terms of the existence of a spanning set of complementary connections in the formula, that is a direct consequence of the previous result.

Theorem 6.3 *An ALL sequent* S *is provable if and only if there exists a set* $\mathcal{SC}$ *of complementary connections that span* S*.*

6.3 An example

Let us come back to the example. The construction of the paths taking into account the α and β types (see figure 6) leads to the following atomic paths:

(1) $\{a_2, a_4, a_{12}, a_{13}, a_{17}, a_{18}\}$ with the connection (a_4, a_{17}),
(2) $\{a_2, a_4, a_{15}, a_{16}, a_{17}, a_{18}\}$ with the connection (a_4, a_{17}),
(3) $\{a_2, a_6, a_7, a_{12}, a_{13}, a_{17}, a_{18}\}$ with the connection (a_6, a_{12}),
(4) $\{a_2, a_6, a_7, a_{15}, a_{16}, a_{17}, a_{18}\}$ with the connection (a_7, a_{15}).

Consequently we have the set of connections $\mathcal{SC} = \{(a_4, a_{17}), (a_6, a_{12}), (a_7, a_{15})\}$.
Now we have to calculate the prefixes for each connection (underlying the positions of special types Φ).
- for the connection (a_4, a_{17}), we have $pre(a_4) = \overline{a_1}\ a_3\ a_4$ and $pre(a_{17}) = \overline{a_8}\ \overline{a_9} a_{17}$ and we deduce a substitution σ such that $\sigma(a_3) = \overline{a_9}$ and $\sigma(a_{17}) = \overline{a_4}$.

- for the connection (a_6, a_{12}), we have $pre(a_6) = \overline{a_1}\ a_3\ \overline{a_5}\ a_6$ and $pre(a_{12}) = \overline{a_8}\ \overline{a_9}\ a_{10}\ \overline{a_{11}}\ a_{12}$ and we deduce a substitution σ such that $\sigma(a_3) = \overline{a_9}$, $\sigma(a_{10}) = \overline{a_5}$ and $\sigma(a_{12}) = \overline{a_6}$.

- for the connection (a_7, a_{15}), we have $pre(a_7) = \overline{a_1}\ a_3\ \overline{a_5}\ a_7$ and $pre(a_{15}) = \overline{a_8}\ \overline{a_9}\ a_{10}\ \overline{a_{14}}\ a_{15}$ and we deduce a substitution σ such that $\sigma(a_3) = \overline{a_9}$, $\sigma(a_{10}) = \overline{a_5}$ and $\sigma(a_{15}) = \overline{a_7}$.

These substitutions are complementary for all the connections and then the sequent is provable in ALL. Moreover the reduction ordering given by (1) $a_3 \to a_9$, (2) $a_{10} \to a_5$ captures constraints for the sequent proof construction. We have to reduce the formula a_3 before the formula a_9 and the formula a_{10} before the formula a_5.

6.4 A connection method for ALL

From this connection characterization we can try to propose a connection method for the ALL fragment. Because of the similarities with the case of intuitionistic logic [48] a first possibility could be to directly adapt the design of the connection method described in [42]. In this context, we will study the relationship with similar or different methods in intuitionistic and other non-standard logics [13]. Having in mind that the connections can be interpreted as semantic constraints generated from the analysis of the sequent, we could compare this work with the approach of [47] that gives polynomial time decision procedure for intuitionistic logic, with notions as proof skeletons and constraints. Moreover the research of adequate substitutions involves the use of algorithms for an efficient string unification.
A second possibility could be as in the MLL case, to use an algorithm for proof nets construction as basis of your proof-search method. Therefore we need to have the right notion of ALL proof net and to define such an algorithm to construct such a proof net, from a given ALL formula. In fact an ALL proof net can be clearly defined as a restriction to ALL of the notion of MALL proof net, recently proposed by J.Y. Girard in [30]. Moreover we could use algorithms as the one defined with B. Martin for automated proof net construction in the MALL fragment [21]. It returns the corresponding proof net for a given provable sequent in ALL (particular restriction of MALL) and then as for MLL could provide a procedure to search the set of connections justifying the provability.

7 Conclusion

We have fixed the relationships between connection based proof-search methods and proof nets construction in MLL. In fact, a procedure for proof nets construction can directly lead to a connection method. We have illustrated this result with a known algorithm [24, 20] from which we can derive a connection-based proof-search method that can be seen also as a top-down proof-search procedure in linear sequent calculus. Independently, [35] has recently presented a characterization of logical validity in MLL (see also [15]) and a connection-based proof-search procedure that is a straightforward extension of a method originally developed for classical and intuitionistic logics [42]. Comparisons between both methods would be interesting even if they have strongly different theoretical foundations. The latter is an extension of existing methods and leads to a uniform approach adapted to various logics, completed with a procedure for sequent reconstruction [44]. Our approach focus on the use of proof net, specific concept of LL, as a tool for automated deduction. It can be naturally seen as an efficient procedure to construct, from a given sequent, a proof net (or all proof nets) and also the sequent proof in a top-down approach, avoiding conversion procedures. Moreover it also corresponds to another alternative solution to the problem of sequent reconstruction from a proof net [6]. Having improved this approach of automated deduction in LL fragments based on proof nets construction, we expect that the underlying principles of construction would also be helpful in the case of (semantically) labelled logical systems. Such possible connections will be studied in further work. From these results we could expect to consider the MALL fragment by connecting provability, proof nets construction and sequent proofs construction in a similar way. The presentation of the explicit relationships between some concepts of proof-search and other logical concepts related to the proof net notion method is an interesting result from both points of view.

References

[1] 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.

[2] V. Alexiev. Applications of Linear Logic to Computation: An Overview. *L. J. of the IGPL*, 2(1):77–107, 1994.

[3] J.M. Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

[4] P. Andrews. Theorem proving via general mating. *Journal of ACM*, 28(2):193–214, 1981.

[5] 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.

[6] R. Banach. Sequent reconstruction in LLM - a sweepline proof. *Annals of Pure and Applied Logic*, 73:277–295, 1995.

[7] G. Bellin. Proof nets for multiplicative and additive linear logic. Technical Report ECS-LFCS 91-161, Department of Computer Science, Edinburgh University, May 1991.

[8] G. Bellin and J. Ketonen. A decision procedure revisited: Notes on direct logic, linear logic and its implementation. *Theoretical Computer Science*, 95:115–142, 1992.

[9] W. Bibel. On matrices with connections. *Journal of ACM*, 28(4):633–645, 1981.

[10] W. Bibel. A deductive solution for plan generation. *New Generation Computing*, 4:115–132, 1986.

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

[12] G.V Davydov, S. Yu. Maslov, G.E. Mints, V.P. Orevkov, and A. O. Slisenko. A computer algorithm for the determination of deducibility on the basis of the inverse method. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning (Classical papers on Computational Logic), volume 2*, pages 531–541. Springer Verlag, 1983.

[13] R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *Journal of Symbolic Logic*, 57:795–807, 1992.

[14] J.C. Filliâtre. A decision procedure for direct predicate calculus - study and implementation in the Coq system. Research report 96-25, LIP - Ecole Normale Supérieure de Lyon, 1995.

[15] B. Fronhöfer. *The Action-as-Implication Paradigm.* CS Press, München, 1996.

[16] D. Galmiche. Proof search methods in linear logic. In *Workshop on Theorem Proving with Analytic Tableaux and related Methods*, Marseille, April 1993.

[17] D. Galmiche. Connection methods and proof nets construction in linear logic fragments (abstract). In *CADE-13 Workshop on Proof Search in Type-Theoretic Languages*, New Brunswick, NJ, USA, July 1996.

[18] D. Galmiche and J.Y. Marion. Improving management of additives for efficient proof search. Unpublished Draft, November 1995.

[19] D. Galmiche and J.Y. Marion. Semantic proof search methods for ALL - a first approach -. In *4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, St Goar am Rhein, Germany, May 1995.

[20] D. Galmiche and B. Martin. Automated proof nets construction in multiplicative and additive linear logic. Technical report, CRIN-CNRS, 1997.

[21] D. Galmiche and B. Martin. Proof search and proof nets construction in linear logic. In *4th Workshop on Logic, Language, Information and Computation, Wollic'97*, Fortaleza, Brasil, August 1997. Logic Journal of IGPL, vol. 5-6, pp 883-885.

[22] D. Galmiche and B. Martin. Proof nets construction and automated deduction in non-commutative linear logic - extended abstract. In *CADE Workshop on Proof search in type-theoretic languages*, pages 51–66, Lindau, Germany, July 1998. to appear in Electonic Notes in Theoretical Computer Science, volume 17.

[23] D. Galmiche and G. Perrier. Automated deduction in additive and multiplicative linear logic. In *Logic at Tver '92, Logical Foundations of Computer Science Symposium, LNCS 620*, pages 151–162, Tver, Russia, July 1992.

[24] 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.

[25] 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.

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

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

[28] J.Y. Girard. Geometry of interaction III: accommodating the additives. In J.Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 329–389. Cambridge University Press, 1995.

[29] 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.

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

[31] 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.

[32] M. Kanovich. Linear logic as a logic of computations. *Annals of Pure and Applied Logic*, 67:183–212, 1994.

[33] J. Ketonen and R. Weyhrauch. A decidable fragment of predicate calculus. *Theoretical Computer Science*, 32:297–307, 1984.

[34] N. Kobayashi and A. Yonezawa. Type-theoretic foundations for concurrent object-oriented programming. In *ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications, OOPSLA'94*, pages 31–45, 1994.

[35] C. Kreitz, H. Mantel, J. Otten, and S. Schmitt. Connection-based proof construction in linear logic. In *14th Int. Conference on Automated Deduction*, pages 207–221, Townsville, North Queensland, Australia, 1997.

[36] V. Lifschitz. What is the inverse method? *Journal of Automated Reasoning*, 5(1):1–24, 1989.

[37] 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.

[38] P. Lincoln and N. Shankar. Proof search in first-order linear logic and other cut-free sequent calculi. In *9th IEEE Symposium on Logic in Computer Science*, pages 282–291, Paris, France, July 1994.

[39] J.Y. Marion. Additive linear logic with analytic cut. In *3rd Workshop on Logic, Language, Information and Computation, WoLLIC'96*, Salvadore, Brazil, May 1996. Journal of IGPL, 4:3, 511-513.

[40] D. Miller. Forum: A multiple-conclusion specification logic. *Theoretical Computer Science*, 165(1):201–232, 1996.

[41] G. Mints. Resolution calculus for the first order linear logic. *Journal of Logic, Language and Information*, 2:59–83, 1993.

[42] J. Otten and C. Kreitz. A connection based proof method for intuitionistic logic. In *4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods, LNAI 918*, pages 122–137, St Goar am Rhein, Germany, 1995. Springer Verlag.

[43] H. Schellinx. Some syntactical observations on linear logic. *Journal of Logic and Computation*, 1(4):537–559, 1991.

[44] S. Schmitt and C. Kreitz. On transforming intuitionistic matrix proofs into standard sequent proofs. In *4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods, LNAI 918*, pages 106–121, St Goar am Rhein, Germany, 1995. Springer Verlag.

[45] S. Schmitt and C. Kreitz. Converting non-classical matrix proofs into sequent-style systems. In *13th Int. Conference on Automated Deduction, LNAI 1104*, pages 418–432, New Brunswick, NJ, USA, 1996.

[46] T. Tammet. Proof strategies in linear logic. *Journal of Automated Reasoning*, 12:273–304, 1994.

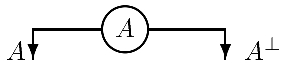
[47] A. Voronkov. Proof-search in intuitionistic logic based on constraint satisfaction. In *5th Int. Workshop TABLEAUX'96, LNAI 1071*, pages 312–327, Terrasini, Palermo, Italy, May 1996.

[48] L.A. Wallen. *Automated Proof search in Non-Classical Logics.* MIT Press, 1990.

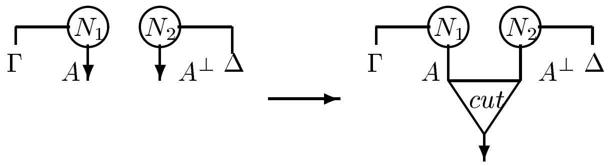
A Inductive MLL proof net

Taking into account that our aim is to have efficient algorithms to construct directly and automatically proof nets, it appears interesting to use an inductive definition of MLL proof nets (similar to the one proposed in [24]).

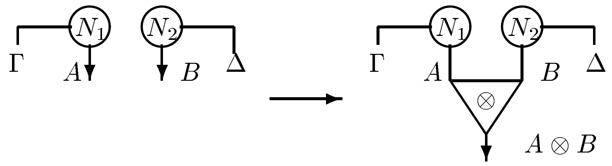
Definition A.1 *(MLL proof net) A MLL (commutative) proof net (PN) is defined as a graph by induction as follows:*
(0) Axiom. *We draw an axiom link, corresponding to* $\vdash A, A^{\perp}$ *as follows and so we obtain an elementary PN with* $A, A^{\perp}$ *as conclusions.*



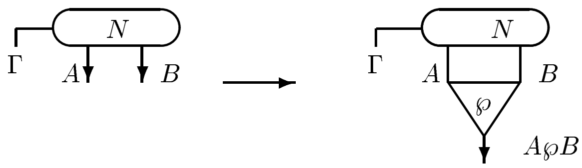
(1) Cut. *If* N_1 *and* N_2 *are disjoint PN's with the multi-sets* Γ, A *and* $A^{\perp}, \Delta$ *respectively as conclusions, we draw a cut-link as follows and so we obtain a PN with the multi-set* Γ, Δ *as conclusions.*



(2) Times. *If* N_1 *and* N_2 *are disjoint PN's with the multi-sets* Γ, A *and* B, Δ *respectively as conclusions, we draw a times link as follows and so we obtain a PN with the multi-set* $\Gamma, A \otimes B, \Delta$ *as conclusions.*



(3) Par. *If* N *is a PN with the multi-set* $\Gamma, A, B,$ *as conclusions, we draw a par link as follows and so we obtain a PN with the multi-set* $\Gamma, A \wp B$ *as conclusions.*



In fact, this definition corresponds to the one of inductively generated proof-structures (IPS) in MLL^{-} for which it is proved that a IPS is a proof net [7].