Electronic Notes in Theoretical Computer Science 37 (2000)
URL: http://www.elsevier.nl/locate/entcs/volume37.html

Proof-search and Proof nets in Mixed Linear Logic

D. Galmiche and J.M. Notin

LORIA UMR 7503 - UHP Nancy 1
Campus Scientifique, BP 239
Vandœuvre-lès-Nancy, France.
E-mail: galmiche{notin}@loria.fr

Abstract

In this paper we study how to design proof-search methods for the multiplicative fragment of Mixed Linear Logic which combines both commutative and non-commutative connectives. After an analysis of the interaction and relationships between these connectives, we propose two different proof-search algorithms, both based on the construction of labelled proof nets. The labels allow in the former to fix some constraints to satisfy during the proof net search and in the latter to propagate some informations useful to guide and control the search. Such based-on proof nets methods are useful to efficiently detect the non-provability and could lead to the generation of some countermodels.

Introduction

Many works have been recently devoted to the study of linear logic [12] (or some of its fragments) and its ability, as a logic of resources, to be easily and strongly related with different topics in computer science, like process calculi or concurrent constraint programming [3]. In many cases, it appears that commutative linear logic is not enough expressive to specify processes having both asynchronous and synchronous behaviors and thus a possible solution can be non-commutative logic [2] or non-commutative connectives to deal with sequentiality. P. Ruet has proposed a mixed version of classical propositional linear logic which combines both commutative and non-commutative connectives [1]. The main motivation of such a proposal is the definition of logical characterizations of synchronization mechanisms in concurrent constraint programming [18]. But to apply such a logic to process calculi and concurrent programming (specification, verification) it is necessary to study appropriate methods and tools dedicated to proof-search. The multiplicative fragment of

Mixed Linear Logic [1] (MMLL) extends the commutative multiplicative linear logic (MLL) and the non-commutative (or cyclic) multiplicative linear logic (NCMLL). Therefore our previous works on proof-search in both fragments that are based on proof nets [8,9] and some graph-theoretic characterizations of provability [16] are natural starting points for our study dedicated to MMLL. A first step consists in understanding and analyzing the interaction of the commutative and non-commutative connectives during the proof-search process. It leads to possible graph-theoretic characterizations of provability from which we try to design based-on labels proof systems for NCMLL. The interest of labelled proof systems is illustrated with the design of such a system for NCMLL with constraints on labels that reflect the characterization and the algorithm we have previously proposed [9,16].

From the initial definitions of proof structures and proof nets in MMLL, we analyze the construction of such structures in two dual ways: i) to consider MMLL from NCMLL, with an interpretation of the commutative connectives into the corresponding non-commutative connectives and ii) to consider MMLL from MLL, with an interpretation of the non-commutative connectives into the corresponding commutative connectives. Taking into account our initial works on proof-search in MLL and NCMLL, we focus on the construction of labelled proof nets to analyze provability or non-provability. The role of labels is in both cases to translate in a simple and useful way the corresponding graph-theoretical characterizations. Such labels include enough semantical information to be used during the proof-search process that is controlled or guided by the satisfaction of constraints. Thus, we design two different algorithms for proof nets construction in MMLL, that are based on particular managements of labels. It means that, in the context of the above mentioned applications, we can expect to design practical and efficient proof-search methods for interesting fragments of Mixed Linear Logic. Even if linear logic and some of its proof-theoretical concepts like proof nets firstly could appear far from practice, recent works about use of linear logic as specification logics [7,15] or about proof-search methods in this logic, for instance based on connections [8,14], have clarified its potential practical uses. The definition of labels and of corresponding constraints for the search process is a natural step to translate some proof-theoretical results of useful fragments of linear logic, like MMLL, into simple and tractable automated proof-search procedures, that are based on the construction of semantical structures like proof nets [13].

[1] In [1], this logic is called non-commutative linear logic. Here we keep the initial appellation of mixed linear logic to avoid some confusions.

$$\frac{}{\vdash A^\perp, A}\ ax \qquad \frac{\vdash \Gamma, A \quad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta}\ cut$$

$$\frac{\vdash \Gamma[\Delta;\Sigma]}{\Gamma[\Delta,\Sigma]}\ entr \qquad \frac{\vdash \Gamma, \Delta}{\vdash \Gamma;\Delta}\ seesaw$$

$$\frac{\vdash \Gamma; A \qquad \vdash B;\Delta}{\vdash \Gamma; A \odot B;\Delta}\ \odot \qquad \frac{\vdash \Gamma; A; B}{\vdash \Gamma; A \lhd B}\ \lhd$$

$$\frac{\vdash \Gamma, A \qquad \vdash B,\Delta}{\vdash \Gamma, A \otimes B,\Delta}\ \otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A ⅋ B}\ ⅋$$

Fig. 1. The MMLL sequent calculus

1 Mixed Linear Logic

In this section we consider the multiplicative fragment of mixed linear logic, called MMLL, that is an extension of both commutative (MLL) and non-commutative or cyclic (NCMLL) linear logic [17,18]. This logic is an alternative to commutative classical, intuitionistic or linear logics for applications in linguistics or in computer science (for instance process calculi or concurrent constraint programming [18]).

1.1 The sequent calculus

The formulae (of MMLL) are build from literals $p, q, \ldots, p^\perp, q^\perp, \ldots$ with commutative connectives (conjunction $\otimes$ and disjunction ⅋). and non-commutative connectives (conjunction $\odot$ and disjunction $\lhd$). The negation is defined as follows with the De Morgan laws: $(p)^\perp = p^\perp$, $(p^\perp)^\perp = p$, $(A \odot B)^\perp = B^\perp \lhd A^\perp$, $(A \lhd B)^\perp = B^\perp \odot A^\perp$, $(A \otimes B)^\perp = B^\perp ⅋ A^\perp$, $(A ⅋ B)^\perp = B^\perp \otimes A^\perp$. The formulae of MLL (resp. NCMLL) are built from the atoms $p, q, \ldots, p^\perp, q^\perp, \ldots$ and the connectives $\otimes$ and ⅋ (resp. $\odot$ and $\lhd$). In order to mix commutative and non-commutative connectives in the a same sequent calculus, two separators of formulae are used in a sequent: a commutative one ',' and a non-commutative one ';'. In the MMLL sequent calculus (see figure 1), some explicit structural rules define the relationships between these separators.

The notation $\Gamma[\ ]$ represents a context with a hole in its decomposition tree and $\Gamma[\Delta]$ is the context $\Gamma[\ ]$ where the hole is filled by Δ. The fundamental results about MMLL (cut-elimination, subformula property, phase semantics, completeness theorem) are given in [17]. Let us mention that there also exists a sequent calculus with only a commutative separator and implicit structural rules [1,17].

Let us consider the sequent $\vdash C \odot (D \odot A), A^\perp ⅋ B^\perp, B \otimes (D^\perp \lhd C^\perp)$ and the following proof in the MMLL sequent calculus

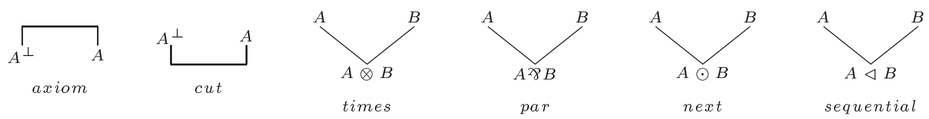
$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$\vdash B^\perp, B$}
\AxiomC{}
\UnaryInfC{$\vdash C^\perp, C$}
\AxiomC{}
\UnaryInfC{$\vdash D^\perp, D$}
\AxiomC{}
\UnaryInfC{$\vdash A^\perp, A$}
\BinaryInfC{$\vdash D^\perp, D \odot A, A^\perp$}
\UnaryInfC{$\vdash D \odot A, A^\perp, D^\perp$}
\BinaryInfC{$\vdash C^\perp; C \odot (D \odot A); A^\perp; D^\perp$}
\UnaryInfC{$\vdash C \odot (D \odot A); A^\perp; D^\perp; C^\perp$}
\UnaryInfC{$\vdash C \odot (D \odot A); A^\perp; D^\perp \lhd C^\perp$}
\UnaryInfC{$\vdash C \odot (D \odot A), A^\perp, D^\perp \lhd C^\perp$}
\UnaryInfC{$\vdash D^\perp \lhd C^\perp, C \odot (D \odot A), A^\perp$}
\BinaryInfC{$\vdash B^\perp, B \otimes (D^\perp \lhd C^\perp), C \odot (D \odot A), A^\perp$}
\UnaryInfC{$\vdash C \odot (D \odot A), B \otimes (D^\perp \lhd C^\perp), A^\perp, B^\perp$}
\UnaryInfC{$\vdash C \odot (D \odot A), B \otimes (D^\perp \lhd C^\perp), A^\perp \parr B^\perp$}
\UnaryInfC{$\vdash C \odot (D \odot A), A^\perp \parr B^\perp, B \otimes (D^\perp \lhd C^\perp)$}
\end{prooftree}
$$

If we replace $D^\perp \lhd C^\perp$ by $C^\perp \lhd D^\perp$ in the initial sequent, then there is no possibility, from the sequent $\vdash D^\perp; C \odot (D \odot A); A^\perp; C^\perp$, to apply structural rules to recover a sequent form for which the $\odot$ rule can be applied.
If we replace, in the initial sequent, $C \odot (D \odot A)$ by $C \otimes (D \odot A)$, the proof construction stops with the sequent $\vdash C^\perp; C \otimes (D \odot A); A^\perp; D^\perp$, that cannot be transformed into the sequent $\vdash C^\perp, C \otimes (D \odot A), (A^\perp; D^\perp)$ and thus the $\otimes$ rule cannot be applied.
From a first analysis of proof-search with this sequent calculus, we observe that the main problems arise because some orders between formulae are fixed by the non-commutative connectives and moreover the commutative and non-commutative connectives interact in the same framework. It also appears that an approach of proof-search in MMLL based on the notion of proof net could be a fruitful alternative in order to avoid the explicit use of the structural rules during the proof-search process.

1.2 Proof nets in Mixed Linear Logic

We recall in this section the initial definition of a proof net in MMLL [1,17] that is based on the notion of trip in a proof structure.

Definition 1.1 *The* links *of MMLL are the following graphs, the vertices of which are labelled with MMLL formulae:*



The axiom-links have two conclusions and no premise and the cut-links have two premises and no conclusion. In the other cases, A and B are respectively the first and second premise and the third formula is the conclusion.

Definition 1.2 *A* proof structure *(of MMLL) is a graph built from links of MMLL such that every occurrence of formula is either the conclusion of exactly one link or the premise of at most one link. An occurrence that is not a premise of a link is called* terminal *and it is a* conclusion *of the structure.*

A proof structure of MLL (resp. NCMLL) is a proof structure labelled with only MLL (resp. NCMLL) links.

Definition 1.3 *Let π be a MMLL proof structure, for any link l of π, a* switching position *is an element of a set $S(l)$ of partial functions (see figure 2). Any function s such that, for any l of π, $s(l) \in S(l)$, is called a* switching *for π. A switching for π is* $\lhd_3$-free *if for every $\lhd$-link, $s(l) \neq \lhd_3$.*

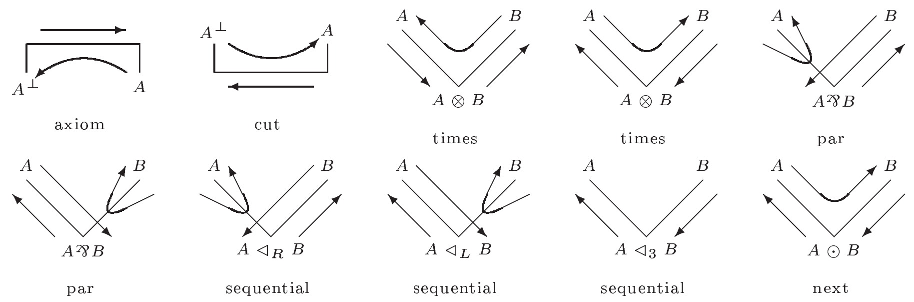


Fig. 2. Switching positions for MMLL

Definition 1.4 *Let π be a proof structure and s be a switching for π, then $s(\pi)$ is the oriented graph with nodes being the decorated formulae of π and with an oriented edge from A^x to B^y if and only if either there exists a link l such that $B^y = s(l)(A^x)$, or we have $A^x = C^\downarrow$ and $B^y = C^\uparrow$ and C is a conclusion of π. A* trip *is defined as a cycle or a maximal path in $s(\pi)$.*

Definition 1.5 *The* inner part *of a link $A \lhd B$ in $s(\pi)$ is the trip from $B^\uparrow$ to $A^\downarrow$.*

Two inner parts do not overlap if either one is included into the other, or they are disjoint [1].

Definition 1.6 *Let π be a proof structure and $s(\pi)$ a switching for π, a trip v in $s(\pi)$ is said* long *if v is a cycle and any occurrence of formula A appears twice: one time decorated with $\downarrow$, another time with $\uparrow$. A cycle v in $s(\pi)$ is said* bilateral *if v does not have the form $A^x, .., B^y, .., A^{\overline{x}}, .., B^{\overline{y}}, .., A^x$ where A and B are occurrences of formulae in π.*

Theorem 1.7 *([1]) Let π be a proof structure of MMLL, then π is a proof net if and only if for any switching s of π we have: i) there exists exactly one cycle σ in $s(\pi)$, ii) σ contains all the conclusions and iii) σ is bilateral.*

This correction criterion is difficult to directly apply to the analysis of provability in MMLL and even for proof net checking. But we aim at focusing on proof-search and mainly on proof net construction in this logic. For that, we have to propose, if possible, appropriate graph-theoretic characterizations of provability in MMLL and then to derive tractable and efficient proof-search procedures. We now remind that the previous criterion for MMLL prof nets correctness is equivalent to another criterion based on the correctness for MLL proof nets and on the satisfaction of conditions on the inner parts of the $\lhd$-links.

Theorem 1.8 *([1]) Let π be a MMLL proof structure, π is a MMLL proof net if and only if its commutative translation π^* (obtained by replacing in π $\odot$ by $\otimes$ and $\lhd$ by $⅋$) is a proof net of MLL such that for every $\lhd_3$-free switching s for π, the inner parts of $\lhd$-links in $s(\pi)$ contain no conclusion and do not overlap.*

Let us note that we can define a proof net of MLL (resp. NCMLL) as a proof structure of MLL (resp. NCMLL) that is a proof net of MMLL. Then we can recover the based-on trip characterization of MLL proof net, that means that a proof structure π is a proof net if and only if for every switching s for π there is a long trip in $s(\pi)$ [12].

2 Commutative and non-commutative proof nets

Let us start to recall some results about graph-theoretic characterizations of provability in MLL and NCMLL and their relationships with proof nets construction.

2.1 Commutative proof nets

An alternative and useful graph-theoretic characterization of MLL proof nets has been proposed in [6] with the following definition:

Definition 2.1 *A* directed Danos-Regnier graph *(D-R graph) is a directed graph which consists of axioms-links (with two out-edges), cut-links (with two in-edges), $\otimes$-links and $⅋$-links (with two in-edges and one out-edge) and also conclusions nodes.*

Theorem 2.2 *([6]) A D-R graph is a proof net of MLL if and only if it is always acyclic and connected under any choice of par-switching (switching condition)*[2] *.*

This characterization can also be seen in a complementary way. A proof structure is in fact a decomposition tree of the sequent (or formula) to prove with the addition of axiom-links (or connections). A proof net is such a proof

[2] It means that one selects a premise of a $⅋$-link and suppresses the edges corresponding to the other premise.

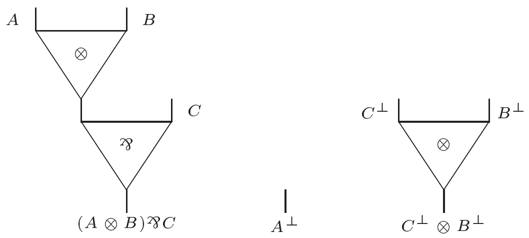
structure but with appropriate axiom-links that characterize it as a representation of legal proofs. Then relationships between based-on connection methods in MLL [14] and proof nets construction methods naturally arise [8].

Let us recall the principles of our algorithm for MLL proof nets construction presented some years ago in [10] and in a revised way in [9]. It is based on a natural concept in MLL, namely the proof net and its inductive definition, from which we observe that from two disjoint proof nets, we can build a new proof net by fusion and extension with a $\otimes$-link and that from a given proof net we can build a new proof net by extension with a $\parr$-link.

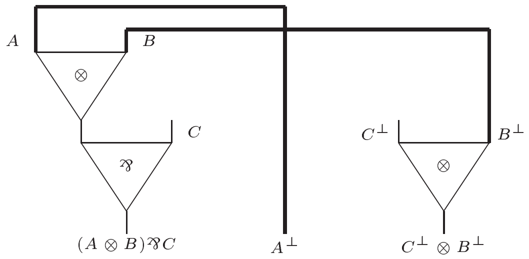
Principles of construction.
Therefore, our algorithm considers the decomposition tree of the given sequent, that will be used as a frame to guide the proof net search. Moreover it manages a set $\mathcal{R}$ of intermediate proof nets that is initialized to the empty set. Then, starting from the literals (or leaves) of the decomposition tree, it builds step by step axiom-links, that are elementary proof nets added to $\mathcal{R}$. When conclusions of proof nets of $\mathcal{R}$ are premises of a $\otimes$-link, they are linked and extended with this link to obtain a new proof net that replaces the two initial ones in $\mathcal{R}$. When conclusions of a given proof net of $\mathcal{R}$ are in fact premises of a $\parr$-link, then it is extended with this link to obtain a new proof net that replaces the initial one in $\mathcal{R}$. If two premises of a $\parr$-link (in the decomposition tree) are conclusions of two different nets in $\mathcal{R}$ then we postpone the extension of such a link. This construction of intermediate proof nets leads to a proof net of the initial sequent if it is provable or else to a failure.
It also appears important to determine, if possible, some orders in the treatment of formulae that are conclusions of subnets (and also premises of links). Moreover, the principles induce some tactics or strategies to fix the choice of free literals during the construction: the *inductive construction* of the subnets (elements of $\mathcal{R}$) allows to avoid the exploration of useless branches and the *fusion principle* means that a priority is to merge disjoint subnets with a $\otimes$-link as soon as possible. Consequently, when a premise of a $\otimes$-link has been connected we next consider the other premise. The above inductive construction corresponds in fact to a top-down proof-search method, starting from axioms (axiom-links) and guided by the initial decomposition tree. With respect to the permutability results in linear sequent calculus [11], we have to always apply, if possible, a $\otimes$ (extension) rule before a $\parr$ (extension) rule.
This algorithm uses specific characteristics of MLL and moreover during the proof search for a given sequent it builds at the same time an appropriate set of connections, a sequent calculus proof and also a proof net of this sequent, if it is provable. Then the method avoids the problem of sequent reconstruction from a proof net [5] or from the connection characterization (i.e. the subset of connections leading to provability) [19].

An example

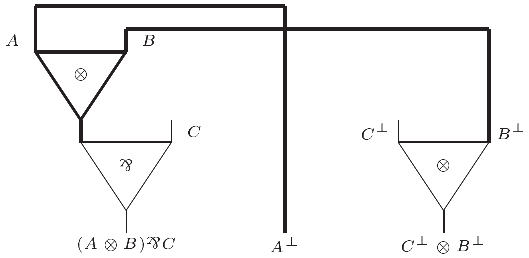
To illustrate these construction principles, we consider the following sequent $\vdash (A \otimes B) \mathbin{⅋} C, A^{\perp}, C^{\perp} \otimes B^{\perp}$. The decomposition tree of this sequent is



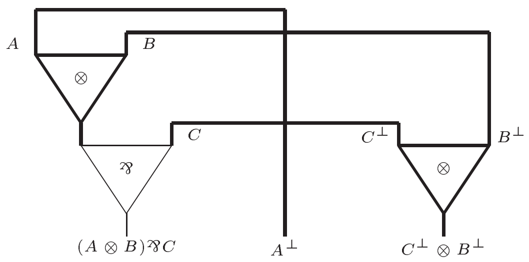
We do not include in this representation the initial ℘-links (linking the different branches or the sequent formulae) but they are implicit. From this decomposition tree, we will construct some new structures (subnets) represented in bold on the figures. Let us notice that we can start with any literal in any branch (formula of the sequent) but a good strategy is to choose one that is a premise of the $\otimes$-link. In this example we have several possible choices and thus we consider the literal A and build the axiom-link (A-$A^{\perp}$). Another strategy consists in dealing with both premises of a $\otimes$-link as soon as possible and consequently we choose B and build the axiom-link (B-$B^{\perp}$).



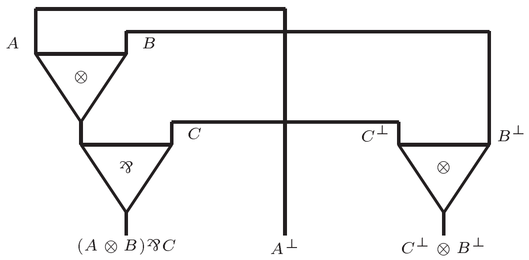
Thus we have two elementary proof nets and they can be connected and extended with the $\otimes$-link to obtain a new partial proof net (in bold in the following figure).



We now choose the free literal that is a premise of the other $\otimes$-link, namely $C^{\perp}$ and then we build the axiom-link ($C^{\perp}$-C) that is a new elementary proof net. Thus, the previous partial proof net and this last axiom-link are connected and extended with the $\otimes$-link to obtain a new partial proof net.



At this step, the previous net is extended with the $\wp$-link because its premises are conclusions of the same partial proof net and thus we obtain the final proof net



and then we can conclude that the sequent is provable in MLL.

2.2 Non-commutative proof nets

But what about the non-commutative linear logic ? The multiplicative fragment of non-commutative linear logic has been defined from MLL by suppressing the exchange rule [2], with the introduction of two negations ${}^\perp A$ and $A^\perp$. If we consider that both are equal, then we obtain a new system [16] that is equivalent to the multiplicative fragment of cyclic linear logic [20].
In the non-commutative logic, there is an effective order between the formulae of a sequent. As a specification logic, it allows to fix an order between actions or states and it has potential applications in planning and verification of concurrent and distributed systems. Non-commutative proof nets can be defined from a notion of long trip as in MLL, but there exists also an adaptation of the D-R graph characterization defined in [16].

Definition 2.3 *A* marked *D-R graph is a D-R graph such that the edges of the $\otimes$ and $\wp$ links are labelled with L (for Left), R (for Right) and C (for Conclusion). A marked D-R graph is* uniformly oriented *if for all $\otimes$ and $\wp$ links the edges labelled R, L et C are oriented in a fixed cyclic order. A marked D-R graph is* strongly planar *if it is plane and uniformly oriented.*

Theorem 2.4 *[16] A marked D-R graph is a proof net if and only if it satisfies the switching condition and is strongly planar.*

This characterization is important from the proof-search point of view and leads to an adaptation (or specialization) to NCMLL of the previous algorithm dedicated to MLL. In this case, from the decomposition tree, one builds axiom-links and subnets like in MLL but with the constraint that the pla-

narity criterion must be satisfied. It means that the graph obtained from the decomposition tree and the axiom-links has to be planar. Consequently, the algorithm for MLL can be adapted in such a way the axiom-links cannot pass each other: the construction of a new axiom-link during the search process is related to the interdiction to cross the existing axiom-links.
If we consider the sequent $\vdash (A \otimes B)⅋C, A^\perp, C^\perp \otimes B^\perp$, the algorithm for NCMLL proceeds as in MLL but it does not allow the construction of the axiom-link (B-$B^\perp$) that must cross the first axiom-link (A-$A^\perp$). Then the algorithm immediately stops with a failure and we deduce that the sequent is not provable in NCMLL. More details about this algorithm for proof nets construction in non-commutative logic and its properties like correction or completeness can be found in [9]. We observe that such a proof-search approach based on proof nets construction is appropriate to detect the non-provability in logics like NCMLL.

2.3 Labelled proof nets

We propose a characterization of the provability in NCMLL which is based on proof nets with static labels and on a particular condition on labels. Let us consider a sequent (or a formula), we build its decomposition tree and then we associate an unique integer n to each literal as a label. This assignment comes from a left-to-right depth first exploration of the decomposition tree, with application of the rules of figure 3. The root of the tree is labelled with 1 and nb_l is the number of literals in the left subtree of the given node. Let us recall that the formulae in a sequent are implicitly linked by ⅋ connectives and that the generation of labels takes this point into account (via the complete decomposition tree).

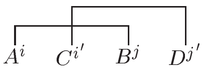
Fig. 3. Labels construction in NCMLL

Let π a proof structure of a sequent S, L_S^π is the set of pairs of labels (i, j) such the literals labelled with i and j are connected with an axiom-link in π.

Theorem 2.5 *Let π be a proof structure of a sequent S, π is a NCMLL proof net if and only if it is a MLL proof net that satisfies the following condition:*

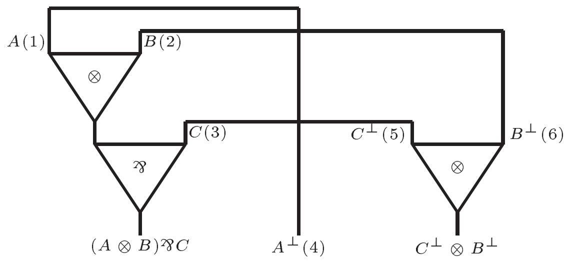
$$\forall (i,j) \in L_S^\pi \ \forall (i',j') \in L_S^\pi \quad i < i' < j \implies i < j' < j \qquad (*)$$

Proof. Let π be a proof structure, if it is not a MLL proof net then it cannot be a NCMLL proof net. Let us assume that π is a MLL proof net and that the condition $(*)$ is not satisfied, then there exists two axiom-links (A-B) and (C-D) such that A, B, C and D have respectively the labels i, j, i' et j' such that: $i < i' < j < j'$. Then the two links have the form



Therefore, from π we cannot build a graph that is strongly planar and consequently π is not a NCMLL proof net. Moreover we can show that if π is not a NCMLL proof net, then there are two axiom-links with the same previous form and then the condition $(*)$ is not satisfied.

We can illustrate this method with the previous example. If we consider the previous MLL proof net then we obtain the following labelled structure:



Thus, we observe that the condition of theorem 2.5 is not satisfied by the axiom-links $(A\text{-}A^\perp)$ and $(B\text{-}B^\perp)$. Such a condition allows to directly verify the non-commutative provability from the commutative provability in MLL. To have labelled structures is also a way to detect and forbid the crossing of axiom-links in the NCMLL algorithm mentioned in the previous section.

2.4 Proof nets in MMLL

Our aim is to propose algorithms for proof-search in MMLL that are based on a graph-theoretic characterization of the provability [16] and then on proof-nets construction. But is it possible to adapt or extend criteria, like those about orientation and planarity used in NCMLL, for a logic where the $\otimes$, $⅋$, $\odot$ and $\lhd$ connectives cohabit ? For instance we observe in MMLL that a proof net can be a graph that is not strongly planar and also that such a planar graph can be associated to a non-valid formula. In fact, the cohabitation of commutative and non-commutative connectives inside MMLL forbid some link crossings and authorize some others in such a graph. As particular case, in MLL, there is no specific restrictions about link crossings and in NCMLL such crossings are forbidden. Therefore, we consider two ways to analyze the provability in MMLL via proof nets

- to study proof-search in MMLL from NCMLL, that corresponds to consider the commutative $⅋$ and $\otimes$ connectives as non-commutative (resp. as $\lhd$ and $\odot$) connectives and to relax some order constraints for these connectives. It means that we can authorize some link crossings in the corresponding non-commutative proof net.
- to study proof-search in MMLL from MLL, that corresponds to consider the non-commutative $\lhd$ and $\odot$ connectives as commutative (resp. as $⅋$ and $\otimes$) connectives and to add some order constraints for these connectives. It

means that we forbid some link crossings in the corresponding commutative proof net.

Before to analyze both approaches we mention a result that emphasizes the specific role of the $\lhd$ connective in MMLL.

Theorem 2.6 *Let π be a MMLL proof structure without $\lhd$-links, π is a proof net iff π^* (commutative version of π* [3] *) is a MLL proof net.*

Proof. By using the characterization of proof nets of MMLL of the theorem 1.8 (see [1]).

A first consequence is that we can build proof nets in the $(\odot, \otimes, \invamp)$ fragment of MMLL by directly using the algorithm defined for MLL. Then for such proof nets there is no planarity condition and link crossings are authorized.

3 Mixed linear logic from non-commutative linear logic

In this section, we consider MMLL from the NCMLL logic and thus we aim at interpreting the commutative $\invamp$ and $\otimes$ connectives as the non-commutative $\lhd$ and $\odot$ connectives, for which we relax some order constraints. We have defined, in section 2.3, a labelled proof system for NCMLL, the labels being an efficient way to attach useful semantical informations in order to guide and control the proof-search process [4]. We aim at characterizing the provability in MMLL, via the notions of proof structure and proof net, and at studying an appropriate notion of planarity for a set of proof structures of NCMLL, that results from a non-commutative interpretation of the commutative connectives in a NCMLL proof structure.

3.1 Preliminary results

Let us first analyze the case when a MMLL proof structure includes some MMLL substructures with two conclusions.

Theorem 3.1 *Let π be a MMLL proof structure and ϕ a MMLL substructure of π with two conclusions, if ϕ is a (sub-) proof net then π is a MMLL proof net if and only if the proof structure π' obtained by replacing ϕ in π with a link (with the two conclusions of ϕ) is a MMLL proof net.*

Theorem 3.2 *Let π be a MMLL proof structure with a node $A \otimes B$, then π is a MMLL proof net if and only if the two proof structures obtained by replacing the node $A \otimes B$ by $A \odot B$ and $B \odot A$ are MMLL proof nets.*

We can now focus on proof nets constructed within the $(\odot, \lhd, \invamp)$ fragment of MMLL. The next result allows to deduce the validity of a structure built with the $\odot$, $\lhd$ and $\invamp$ links, from a corresponding structure built with the $\odot$ and $\lhd$ links.

[3] It means that the $\odot$-links of π are replaced by $\otimes$-links.

Theorem 3.3 *Let π be a proof structure in the $(\odot, \lhd, \parr)$ fragment of MMLL, and $A \parr B$ be a node, then π is a proof net if one of the proof structures obtained by replacing the node $A \parr B$ by $A \lhd B$ or $B \lhd A$ is a MMLL proof net.*

Proof. Let us assume that one of the proof structures obtained by replacing the node $A \parr B$ with $A \lhd B$ or $B \lhd A$ is a proof net. We observe that the switching conditions for $\parr$ are included in the ones for the $\lhd$ connective. Then π is a proof net.

Let us note that this approach does not take into account the associativity of the $\parr$ connective. For instance, in $A \odot (B \odot C)) \parr B^{\perp} \parr (A^{\perp} \parr C^{\perp})$, we observe that we have to find a correct or appropriate representation of $B^{\perp} \parr (A^{\perp} \parr C^{\perp})$ modulo the $\parr$ associativity in order to effectively apply this characterization. Let us summarize the main points: from a MMLL proof structure π of a given sequent, we can generate a set of proof structures in which the $A \otimes B$ nodes have been replaced by $A \odot B$ and $B \odot A$ (this set covers all the combinations of transformation of $\otimes$ nodes) and in which the $A \parr B$ nodes have been replaced by $A \lhd B$ or by $B \lhd A$.
If these proof structures satisfy the conditions of planarity in NCMLL modulo the following two conditions $i)$ and $ii)$, then π is a MMLL proof net.
These conditions are the following: $i)$ if there are subformulae following the $A \parr (B \parr C)$ schema then consider if necessary the set of the possible rewritings modulo $\parr$ associativity and the corresponding decomposition trees; $ii)$ to locally accept some non-planar substructures with two conclusions within π, but in such a way that if we implicitly replace them by fictive links with both conclusions, we obtain a planar structure of NCMLL.

3.2 Labelled structures

As we have analyzed the provability in MMLL by considering all connectives as non-commutative [4] with some authorized link crossings, we aim at defining labelled proof systems where labels allow to simply represent this characterization.
A first step is to recursively build labels [5] within the decomposition tree, following the construction rules of figure 4. The tree root is labelled with 1 and the variables x_n and y_n can only be assigned with 0 and 1. The formulae with commutative connectives can be represented in different equivalent ways: for instance, $A \parr (B \otimes C)$, $(B \otimes C) \parr A$ and $(C \otimes B) \parr A$ are equivalent. Let us note that nb_l (resp. nb_r) represents the number of literals that appear in the left-hand (resp. right-hand) side sub-formulae of the node. From this labelled decomposition tree the goal is to find a set of appropriate connections (between dual literals) in order to obtain a proof structure that is a proof net.

[4] Let us recall that the formulae in a sequent are implicitly connected with $\parr$ connectives and thus we work in MMLL and not only in NCMLL.
[5] Such labels allow, with respect to the provability characterization, to modify the RLC orientation of the nodes $\parr$ and $\otimes$ (cf. definition 2.3).

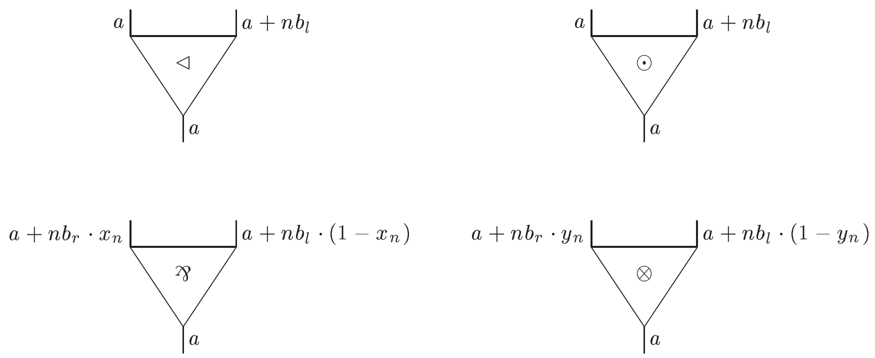


Fig. 4. Labels construction

Having such a procedure to generate labels, the previous graph-theoretic characterization of provability can be translated into the following form:

Theorem 3.4 *Let π be a labelled MMLL proof structure, π is a proof net if for all values of y_i (0 or 1), there exists an assignment of the x_i variables such that the labels satisfy the condition $(*)$ of theorem 2.5, modulo the $⅋$ associativity.*

Proof. The main point is that the change of value of a variable x_i or y_i in π corresponds to inverse the order of the RLC branches of the corresponding node.

3.3 Proof nets construction

The following algorithm is based on the same principles of the algorithm defined in [9] and recalled in section 2.1. What is new here is the use of the static labels that fix some constraints for the application of the extension rules that lead to new subnets, during the construction process. The main steps of the procedure are:

- to build the decomposition tree with labels;
- to build some axiom-links and then some subnets by the fusion and extension rules like in NCMLL (see section 2.2). But in MMLL the application of such rules depends on the satisfaction of conditions about labels, and then of constraints about the variables x_i and y_i.
  - when an $\lhd$ link has to be extended, the constraints arising from the condition of theorem 2.5 (*), must be satisfied. In the other cases, the constraints are memorized but we do not have to satisfy them;
  - each time a subnet with two conclusions is built, we implicitly add a new link with its two (renamed) labelled conclusions. This new link, considered as an extra axiom-link, adds new constraints to satisfy.
- if there is a contradiction ($x_i = 0$ and $x_i = 1$), the algorithm stops. In the other cases, the algorithm has the same behavior as the MLL algorithm.
- if a final proof structure is built, then it is a proof net and the formula is

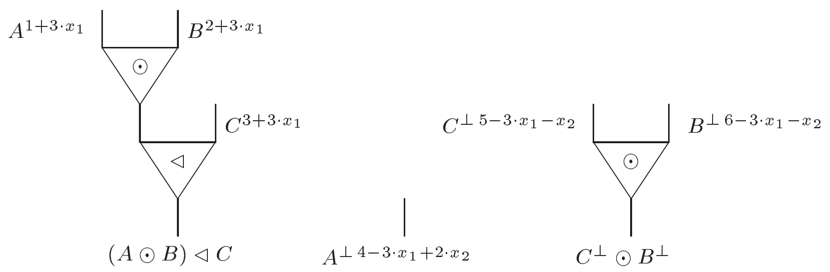
valid. If not, if the sequent to prove includes a $A \parr (B \parr C)$ schema then we recall the algorithm with another rewriting of it modulo the $\parr$ associativity, else the sequent is not valid.

Theorem 3.5 (Correction and completeness) *Let S be a MMLL sequent, this algorithm is correct (if it returns a proof structure for S then it is a proof net and then S is provable in MMLL). Moreover, the algorithm is complete (if S is provable in MMLL then it returns a proof net of S).*

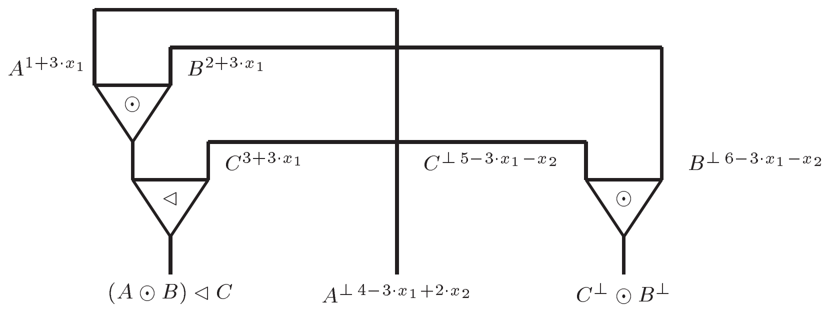
Proof. From the previous results and the corresponding proof for the NCMLL algorithm [9].

3.4 An example

Let us illustrate this algorithm with the sequent $\vdash (A \odot B) \lhd C, A^{\perp}, C^{\perp} \odot B^{\perp}$. It corresponds to the previous example where the commutative connectives are replaced by the corresponding non-commutative connectives. It is important to recall that a comma in a MMLL sequent corresponds to a $\parr$ connective and consequently that we deal with the $(\odot, \lhd, \parr)$-fragment of MMLL. We start with the construction of the following labelled decomposition tree



Let us recall that the formulae of the sequent are effectively linked by $\parr$ connectives into a single formula that is already decomposed. Then the procedure builds the axiom-links $(A^{\perp}, A)$ and $(B^{\perp}, B)$ from the literals A and B and then a new subnet, by the extension rule of the $\odot$ node. Thus, an axiom-link $(C^{\perp}, C)$ is built and the extension rule is again applied to the other $\odot$-link in order to obtain a new subnet. The next step consists in applying the extension rule to the $\lhd$ link that is subject to the following constraints:
$A^{1+3 \cdot x_1} < B^{2+3 \cdot x_1} < C^{3+3 \cdot x_1} < C^{\perp\, 5-3 \cdot x_1-x_2} < B^{\perp\, 6-3 \cdot x_1-x_2} < A^{\perp\, 4-3 \cdot x_1+2 \cdot x_2}$
One can find a solution that is $x_1 = 1$ and $x_2 = 0$. Thus, the last extension rule can be applied and then it leads to a final proof net for the sequent.



If we consider another sequent from the previous one where $C^{\perp} \odot B^{\perp}$ is replaced by $C^{\perp} \otimes B^{\perp}$, then the algorithm follows the same steps till the application of the extension rule to the $\lhd$ connective that depends on the satisfaction of new constraints:
$A^{1+3\cdot x_1} < B^{2+3\cdot x_1} < C^{3+3\cdot x_1} < C^{\perp\, 5-3\cdot x_1-x_2+y_1} < B^{\perp\, 6-3\cdot x_1-x_2-y_1} < A^{\perp\, 4-3\cdot x_1+2\cdot x_2}$.
We show that they cannot be satisfied.
For $y_1 = 0$, $x_1 = 1$ and $x_2 = 0$ is a solution but for $y_1 = 1$ the constraint that is $5 - 3 \cdot x_1 - x_2 + y_1 < 6 - 3 \cdot x_1 - x_2 - y_1$ has no solution. It means that the extension rule cannot be applied to the $\lhd$ node. Then, the previous proof structure where $C^{\perp} \odot B^{\perp}$ is replaced by $C^{\perp} \otimes B^{\perp}$ is not a proof net and moreover we can deduce that this sequent is not provable in MMLL.
Instead of generating general labels in the decomposition tree, we could first build some subnets following the algorithm for NCMLL and only generate simpler labels (and then some constraints), for the subnets including a $\lhd$ link. After the resolution of these constraints, we could go on and implicitly replace the subnet by a fictive link. Such a refinement for the constraints generation has to be analyzed in further works.

4 Mixed linear logic from commutative linear logic

Our aim in this section is to consider MMLL from MLL and the characterization provability given in theorem 1.8 in order to design an algorithm for automatic proof nets construction in MMLL. The first condition explicitly relies on the construction of a MLL proof net, and the second corresponds to some constraints with the non-commutativity of the $\lhd$ and $\odot$ connectives. In fact, we already have an algorithm for automatic proof nets construction in MLL (see [9,10] and section 2.1). Thus, the main idea is to keep the principles of this algorithm and to add enough information on the structure under construction to verify the second condition. For that we propose to define appropriate labels.

4.1 Initialization

A first step consists of the following actions: to build the decomposition tree of the given sequent (or formula), to assign an integer to all the links, and to initialize the labels in the following way

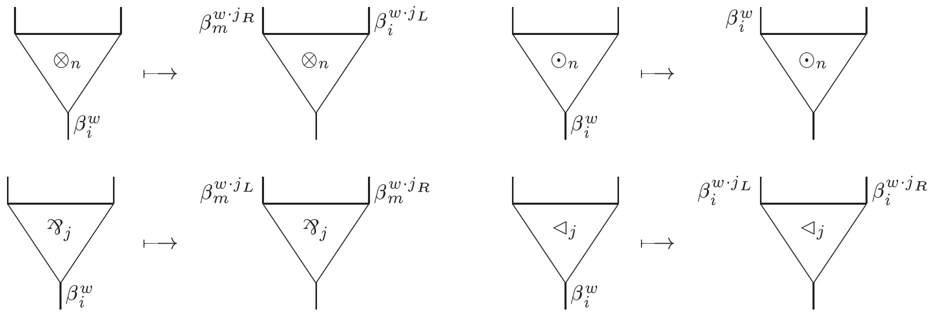
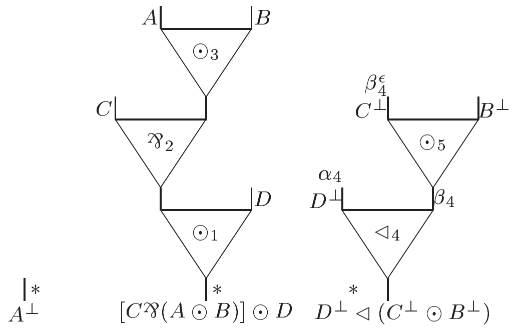


Fig. 5. Rules for label initialization

- each conclusion is marked with $*$,
- for each $\lhd$-node, with the label i, its left and right premise are respectively marked with α_i and β_i or more precisely β_i^ε (ε being the empty word).
- the β_i^ε labels are recursively propagated upward to the literals following the rules for each connective, given in figure 5 [6]. It is important to notice that the initial β_i^ε is not erased in the final labelled tree. We observe that during this initialization the word w attached to β_i^w can be modified by concatenation in some rules.

First, we obtain a decomposition tree with labelled literals. The following figure illustrates the result of the initialization step on the sequent $\vdash A^\perp, (C⅋(A \odot B)) \odot D, D^\perp \lhd (C^\perp \odot B^\perp)$.

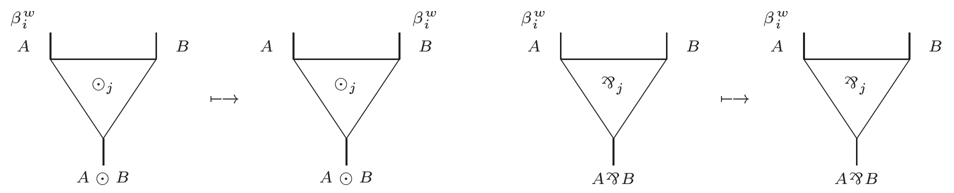


4.2 Label propagation

The algorithm for proof nets construction is based on the search of appropriate axiom-links between dual literals and on the verification of validity with respect to some correctness criteria. To perform such a verification, the labels of type β_i^w attached to literals move through the proof structure under construction, after each construction of an axiom-link, with respect to the propagation rules defined for each kind of link (see appendix A) [7].
Let us precise the main points about the label propagation. If a literal A,

[6] Let us remark that these rules correspond with the switching positions of each link

[7] The way the β_i^w-labels are propagated through the structure is in fact related to the switching positions of the various links (see figure 2)



(a) Example for $\odot$ (b) Example for $\parr$

Fig. 6. Change of direction

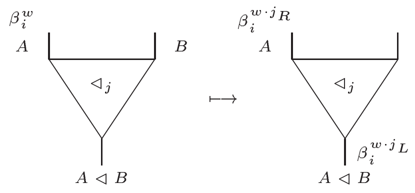
marked with a set of labels of type β_i^w, is connected with its negation $A^\perp$, the labels have to be propagated downward through $A^\perp$ via the axiom-link (see figure A.1). Then, each label recursively moves through the structure beyond $A^\perp$, until it reaches a non-connected literal, a conclusion (marked with $*$), or the left premise of the $\lhd$-link from where it starts (marked with α_i). There are two kinds of rules: the upward rules that are applied when the β-labels moves from conclusion to literals and the downward rules that are applied when the β-labels travel from literals to the conclusion (see appendix A).

An important point about this label propagation is that the direction of propagation of a β_i^w-label can change during its travelling. For example, in figure 6(a), a β_i^w-label moves downward to A and then moves upward from B. In figure 6(b), a β_i^w-label moves downward to A then moves upward from A. During the propagation process the β_i^w-labels travel through the various links. As a β_i^w-label can travel twice through the same link then we have to build a new word for each label that memorizes if the associated label has crossed a given link. Indeed, the way to travel through a link depends if the label has already crossed the link or not. The construction and the use of the word w are described in the rules of the propagation of the β_i^w-label [8].

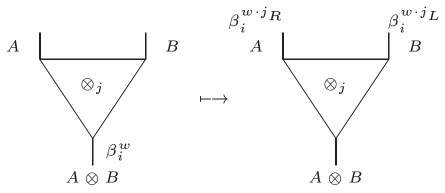
When a β_i^w-label reaches a link of type $\otimes$, $\parr$ or $\lhd$ for the first time, it duplicates into two new labels $\beta_i^{w_1}$ and $\beta_i^{w_2}$ (and β_i^w is erased). The words w_1 and w_2 result from the concatenation of w to a symbol that memorizes that the link has been already crossed by the labels. If $\beta_i^{w_1}$ or $\beta_i^{w_2}$ travel once again through the link, they will not be duplicated anymore. For example, in figure 7(a), a β_i^w-label moves downward to A, duplicates into a $\beta_i^{w\cdot j_L}$-label, which then moves downward from $A \lhd B$, and a $\beta_i^{w\cdot j_R}$-label which then moves upward from A. In figure 7(b), a β_i^w-label moves upward to $A \otimes B$, duplicates to a $\beta_i^{w\cdot j_L}$-label, which then moves upward from B, and a $\beta_i^{w\cdot j_R}$-label, which then moves upward from A.

If a β_i^w-label reaches a free literal or the left premise of the $\lhd$-link from where it starts (labeled with α_i), the propagation stops. If a β_i^w-label reaches

[8] It translates the fact that some kinds of links ($\otimes$, $\parr$ and $\lhd$) have two switching positions (see figure 2).



(a) Example for seq



(b) Example for times

Fig. 7. Label duplication

a conclusion or the right premise of the ⅋-link from where it starts (labeled with β_i), the propagation fails. In fact, the first case ensures that the inner part of the $\lhd_i$-link contains a conclusion, the second ensures that the commutative version of the structure cannot be a MLL proof net. Once a β_i^w-label, originated from the a $\lhd_i$-link, reaches the left premise of this $\lhd$-link (marked with α_i), it can be erased because it will never travel through the structure anymore. In the downward rules for disjunctive links (⅋ and $\lhd$), we observe that the β_i^w-labels remain on the premise from where it comes from. In fact, when a β_i^w-label moves downward through the premise of a disjunctive link, it may have to move upward in the structure.

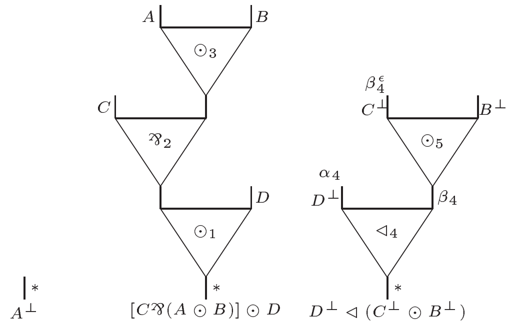
4.3 Principles of the algorithm

This algorithm is based on the algorithm of proof net construction for MLL [10], considering the $\odot$ and $\lhd$ links as respectively $\otimes$ and ⅋ links. Then, it manages the labels, especially the β_i^w-labels, to verify the constraints of non-commutativity for $\odot$ and $\lhd$ links inside MMLL.
The algorithm can be presented with the following steps:
1) The first step of the construction consists in building the decomposition tree that corresponds to the given sequent, and in initializing the labels as described in section 4.1.
2) Then, a free literal, preferably marked with some β_i-label, is selected and then a connection with one of its dual literals is searched. If such a connection is found then the β_i-labels are propagated (see section 4.2).
3) If this propagation stops with a failure then there is a backtracking with a search of another dual literal else the algorithm goes on to search other axiom-links and then (MLL) subnets are built by extension rules like in the MLL case.

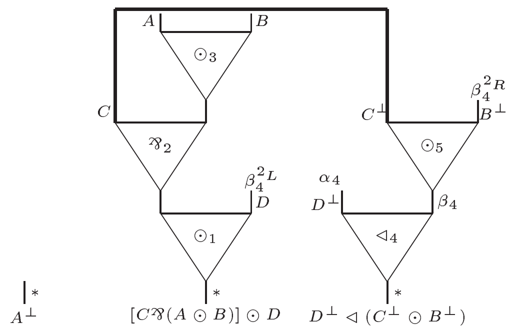
Let us illustrate this algorithm with two examples, one with a valid sequent and another with a non-valid sequent.

4.4 An example

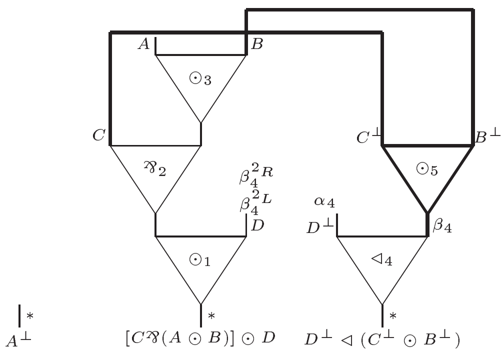
A first example consists in analyzing the provability of the following MMLL sequent $\vdash A^\perp, [C ⅋ (A \odot B)] \odot D, D^\perp \lhd (C^\perp \odot B^\perp)$. Let us begin to build the decomposition tree of this sequent and to initialize it with the labels. Then, we obtain



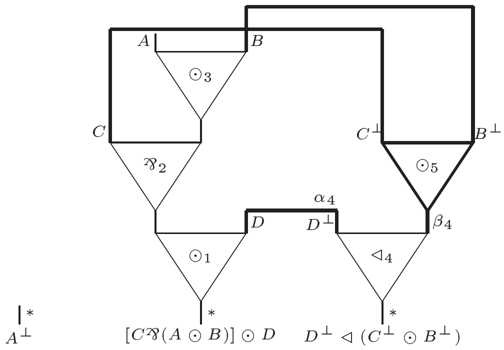
From this labelled structure we aim at building an axiom-link. For that, we choose a literal labelled with a β_i^ϵ; in this example there is only one, namely $C^\perp$, that is labelled with β_4^ϵ. Then, we build the axiom-link (C-$C^\perp$) and we apply propagation rules for this label. At first, β_4^ϵ is propagated in the axiom-link (see rules in figure A.1 in appendix A) and then C is temporally labelled with β_4^ϵ. Because of the $\wp_2$ link, we apply the third rule of figure A.7 and thus we have two β-labels : β_4^{2L} that is propagated downward through the $\odot_1$ link and β_4^{2R} that is retro-propagated from C through the axiom-link and then through the $\odot_5$ link. Therefore we obtain the following structure with new labels.



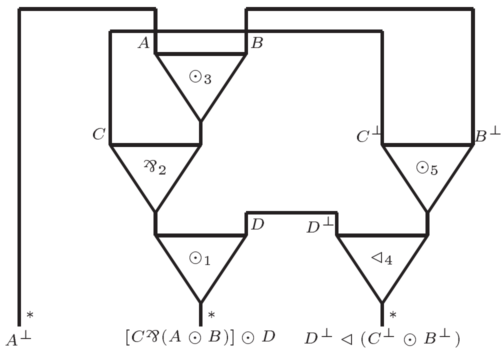
Next, we choose a free literal with a β_i^w-label, that is $B^\perp$ labelled with β_4^{2R}. Then we build the axiom-link (B-$B^\perp$) and we apply the propagation rules for this label. Thus, β_4^{2R} moves downward through the $\odot_3$-link and then through the $\wp_2$-link a second time, but there is no need for a new duplication of the label and it moves downward to the conclusion of the $\wp_2$-link and its propagation goes on through the $\odot_1$-link. We observe that finally the literal D has now two labels namely β_4^{2L} and β_4^{2R}. After this label propagation, we observe that the two axiom-links (that are elementary proof nets) can be connected and extended with the $\odot_5$-link to have a new partial proof net.



Next, we choose another free literal labelled with a β_i^w-link, namely D that is labelled with β_4^{2L} and β_4^{2R}. Then, we build the axiom-link (D-$D^\perp$) and we apply propagation rules for the label. Both labels reach the label α_4, and thus they can be erased.

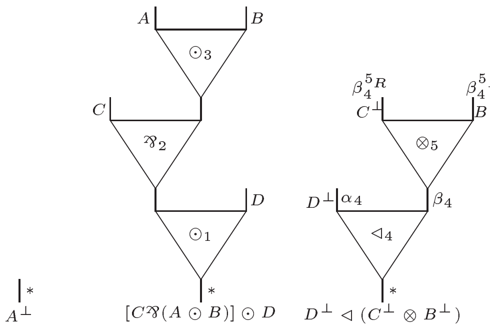


Next, we build the last axiom-link, namely (A-$A^\perp$) for which there is no label propagation. After that, there is no failure because of the labels and we can go on with the construction of the expected proof net. Thus, the previous subnet and this new axiom-link are connected and extended with the $\odot_3$-link. The resulting net is then extended with the $\parr_2$-link. Moreover, this new partial net and the (D-$D^\perp$) axiom-link are extended with the $\odot_1$-link in order to obtain a new partial proof net, that is finally extended with the $\lhd_4$-link. Therefore we have built the following proof net for the given sequent.

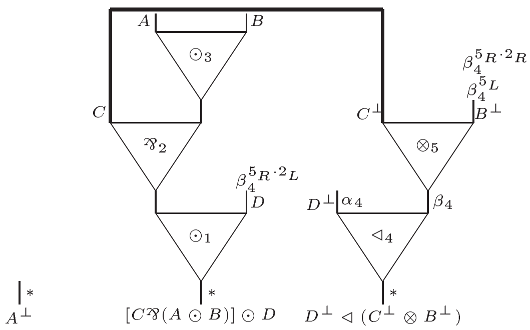


4.5 Another example

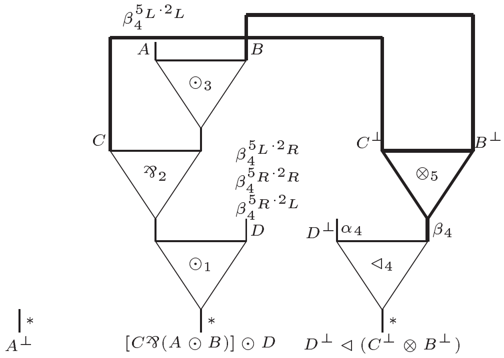
As another example we consider the previous sequent where $C^\perp \odot B^\perp$ is replaced by $C^\perp \otimes B^\perp$, namely $\vdash A^\perp, [C \parr (A \odot B)] \odot D, D^\perp \lhd (C^\perp \otimes B^\perp)$. Thus, we start to build the decomposition tree for this sequent and we initialize the labels. We obtain the following labelled decomposition tree



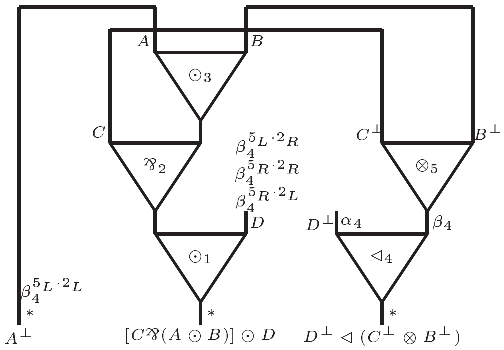
Next, we create the axiom-link (C-$C^\perp$), and we propagate $\beta_4^{5_R}$ in the structure. Let us notice that, unlike in the previous example, we have two β_i^w-labels on $B^\perp$: $\beta_4^{5_L}$ and $\beta_4^{5_R \cdot 2_R}$. From this labelled structure we aim at building an axiom-link. For that, we have to choose a literal labelled with a β_i^w: in this example there are two possibilities, namely $C^\perp$ labelled with $\beta_4^{5_R}$ and $B^\perp$ labelled with $\beta_4^{5_L}$. Then we decide to build the axiom-link (C-$C^\perp$) and we apply the propagation rules for its label. At first, $\beta_4^{5_R}$ is propagated in the axiom-link (rules in figure A.1) and then C is temporally labelled with this label. Because of the $\parr_2$ link, we apply the third rule of figure A.7 and thus we have two β-labels : $\beta_4^{5_R \cdot 2_L}$ that is propagated downward through the $\odot_1$-link and $\beta_4^{2_R}$ that is retro-propagated from C through the axiom-link and then the $\odot_5$-link. Therefore we obtain the following structure with new labels.



Next, we choose a free literal labelled with a β_i^w that is $B^\perp$ labelled with $\beta_4^{5_R \cdot 2_R}$ and $\beta_4^{5_L}$. Thus, we build the axiom-link (B-$B^\perp$) and we apply propagation rules for both labels. The $\beta_4^{5_L}$-label has not yet moved through the $\parr_2$-link and consequently it is duplicated into the labels $\beta_4^{5_L \cdot 2_L}$ and $\beta_4^{5_L \cdot 2_R}$. If we compare this step with the similar step in the previous example, the literal A is now labelled with $\beta_4^{5_L \cdot 2_L}$. We observe that the literal D has finally three labels, namely $\beta_4^{5_L \cdot 2_L}$, $\beta_4^{5_L \cdot 2_R}$, $\beta_4^{5_R \cdot 2_L}$. After this label propagation, we observe that the two axiom-links (that are elementary sub-proof nets) can be extended with the $\otimes_5$-link to have a new partial proof net (of MLL).



For the next step we have a choice between two free literals, namely A and D but, as said before, we always try to choose among the most internal literals in the sequent (or the highest literals in the decomposition tree). Thus, we consider the literal A and build the axiom-link (A-$A^\perp$) with a propagation of its label $\beta_4^{5_L \cdot 2_L}$. Therefore, we obtain the case where a formula already marked with a $*$ also has a β_i^w-label, that is in fact a failure case. Consequently, this last axiom-link construction is not allowed. As it is the only way to connect dual literals, we cannot backtrack and the algorithm finally fails. It is not possible to construct a proof net for the given sequent and then this one is not valid in MMLL.



4.6 Properties of the algorithm

Let us start to prove the following result

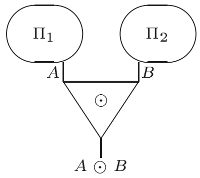
Theorem 4.1 *Let Π be a proof structure of MMLL, Π is a proof net iff*

i) Π^* is a proof net of MLL,

ii) for every $\triangleleft_3$-free switching, the inner parts of $\triangleleft$-links contain no conclusion.

Proof.
$\Rightarrow$ We prove that if Π is a proof net then it verifies the conditions i) and ii). It is immediate from the theorem 1.8.

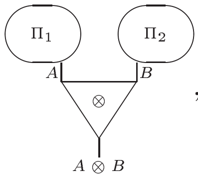
$\Leftarrow$ We prove by structural induction that if Π is a proof structure that verifies the two conditions $i)$ and $ii)$, then it is a proof net.

-  is a proof structure that is a proof net.
- Let Π be a proof structure  such that : Π^* is a proof net of MLL and for every $\lhd_3$-free switching, the inner parts of the $\lhd$-links of Π contain no conclusion.

  Let us prove that Π_1 and Π_2 satisfies the conditions $i)$ and $ii)$.

  It is immediate that Π_1^* and Π_2^* are proof nets of MLL. Moreover Π verifies condition $ii)$, and consequently the inner parts of $\lhd$ links in Π contain no conclusion. In particular, they do not contain $A \odot B$. As Π is a proof net, the trips in Π are of the form : $B^{\downarrow} A \odot B^{\downarrow} A \odot B^{\uparrow} A^{\uparrow} \cdots A^{\downarrow} B^{\uparrow} \cdots B^{\downarrow}$. We can deduce that the inner parts of $\lhd$-links of Π_1 (resp. Π_2) cannot contain A (resp. B), and they cannot contain another conclusion than A (resp. B) because Π verifies the condition $ii)$. Therefore, the inner parts of $\lhd$-links of Π_1 (resp. Π_2) contain no conclusion. Finally, by induction hypothesis, we can deduce that Π_1 and Π_2 are proof nets.

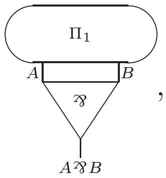
  Let s and t be $\lhd_3$-free switchings for respectively Π_1 and Π_2, then (s,t) is a $\lhd_3$-free switching for Π. Let $\sigma_{1,s}$ (resp. $\sigma_{2,t}$) be the unique cycle in $s(\Pi_1)$ (resp. $t(\Pi_2)$), then : $\sigma_{1,s} = A^{\uparrow} \overset{1}{\cdots} A^{\downarrow}$ and $\sigma_{2,t} = B^{\uparrow} \overset{2}{\cdots} B^{\downarrow}$. Thus, the unique cycle in $(s,t)(\Pi)$ is : $\sigma = B^{\uparrow} \overset{2}{\cdots} B^{\downarrow} A \odot B^{\downarrow} A \odot B^{\uparrow} A^{\uparrow} \overset{1}{\cdots} A^{\downarrow}$.

  As the inner parts of $\lhd$-links of Π_1 (resp. Π_2) contain no conclusion, do not overlap and are entirely included in $\overset{1}{\cdots}$ (resp. $\overset{2}{\cdots}$), we can deduce that the inner parts of $\lhd$-links of Π contain no conclusion and do not overlap. Then, Π is a proof net.
- Let Π be a proof structure ,

  with an argument similar to the $\odot$ case, we can prove that Π_1 and Π_2 both satisfy the conditions $i)$ and $ii)$, and so that Π_1 and Π_2 are proof nets.

  Let s and t be $\lhd_3$-free switchings for respectively Π_1 and Π_2, then $(s,t,\otimes_L)$ and $(s,t,\otimes_R)$ are both $\lhd_3$-free switchings for Π. Let $\sigma_{1,s}$ (resp. $\sigma_{2,t}$) be the unique cycle in $s(\Pi_1)$ (resp. $t(\Pi_2)$), then : $\sigma_{1,s} = A^{\uparrow} \overset{1}{\cdots} A^{\downarrow}$ and $\sigma_{2,t} = B^{\uparrow} \overset{2}{\cdots} B^{\downarrow}$. Thus, the unique cycle in $(s,t,\otimes_R)(\Pi)$ is : $\sigma = B^{\uparrow} \overset{2}{\cdots} B^{\downarrow} A \otimes B^{\downarrow} A \otimes B^{\uparrow} A^{\uparrow} \overset{1}{\cdots} A^{\downarrow}$ and the one in $(s,t,\otimes_L)(\Pi)$ is : $\sigma = A^{\uparrow} \overset{1}{\cdots} A^{\downarrow} A \otimes B^{\downarrow} A \otimes B^{\uparrow} B^{\uparrow} \overset{2}{\cdots} B^{\downarrow}$.

  As the inner parts of $\lhd$-links of Π_1 (resp. Π_2) contain no conclusion, do not overlap and are entirely included in $\overset{1}{\cdots}$ (resp. $\overset{2}{\cdots}$), we can deduce that the inner parts of $\lhd$-links of Π contain no conclusion and do not overlap. Finally, we conclude that Π is a proof net.

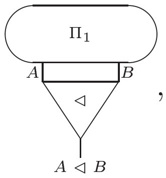
- Let Π be a proof structure 

,

first, we can verify that Π_1^* is a proof net of MLL. Now, we prove that Π_1^* satisfies the condition *ii*). As Π satisfies condition *ii*), we only have to prove that the inner parts of $\lhd$-links of Π_1 contain neither A nor B. Let us suppose that there exists a $\lhd$-link, namely $C \lhd D$, and a switching s for Π_1 such that the inner part of $C \lhd D$ contain A (resp. B). Then, the inner part of $C \lhd D$ in Π for the switching $(s, \parr_L)$ (resp. $(s, \parr_R)$) is : $D^\uparrow \cdots A^\downarrow A \parr B^\downarrow A \parr B^\uparrow A^\uparrow \cdots C^\downarrow$ (resp. $D^\uparrow \cdots B^\downarrow A \parr B^\downarrow A \parr B^\uparrow B^\uparrow \cdots C^\downarrow$). Thus, there exists an inner part of a $\lhd$-link in Π that contains a conclusion $(A \parr B)$, which contradicts the fact that Π satisfies condition *ii*).

Let s be a $\lhd_3$-free switching for Π_1, then $(s, \parr_L)$ and $(s, \parr_R)$ are $\lhd_3$-free switchings for Π. Let $\sigma_{1,s}$ be the unique cycle in $s(\Pi_1)$, then : $\sigma_{1,s} = A^\uparrow \overset{1}{\cdots} B^\downarrow B^\uparrow \overset{2}{\cdots} A^\downarrow$. Then the unique cycle in $(s, \parr_R)(\Pi)$ is : $\sigma_1 = A^\uparrow \overset{1}{\cdots} B^\downarrow A \parr B^\downarrow A \parr B^\uparrow B^\uparrow \overset{2}{\cdots} A^\downarrow$, and the one in $(s, \parr_L)(\Pi)$ is : $\sigma_1 = B^\uparrow \overset{2}{\cdots} A^\downarrow A \parr B^\downarrow A \parr B^\uparrow A^\uparrow \overset{1}{\cdots} B^\downarrow$.

As the inner parts of $\lhd$-links of Π_1 contain no conclusion, do not overlap and are entirely included in $\overset{1}{\cdots}$, we can deduce that the inner parts of $\lhd$-links of Π contain no conclusion and do not overlap. Then, Π is a proof net.

- Let Π be a proof structure 

,

with a similar argument as for the $\parr$ case, we can prove that Π_1 verifies conditions *i*) and *ii*), and then, by induction hypothesis, that Π_1 is a proof net.

Let s be a $\lhd_3$-free switching for Π_1, then $(s, \lhd_L)$ and $(s, \lhd_R)$ are $\lhd_3$-free switchings for Π. Let $\sigma_{1,s}$ be the unique cycle in $s(\Pi_1)$, then : $\sigma_{1,s} = A^\uparrow \overset{1}{\cdots} B^\downarrow B^\uparrow \overset{2}{\cdots} A^\downarrow$. So the unique cycle in $(s, \parr_R)(\Pi)$ is : $\sigma_1 = A^\uparrow \overset{1}{\cdots} B^\downarrow A \lhd B^\downarrow A \lhd B^\uparrow B^\uparrow \overset{2}{\cdots} A^\downarrow$, and the one in $(s, \parr_L)(\Pi)$ is : $\sigma_1 = B^\uparrow \overset{2}{\cdots} A^\downarrow A \lhd B^\downarrow A \lhd B^\uparrow A^\uparrow \overset{1}{\cdots} B^\downarrow$

Let $E \lhd F$ be a $\lhd$-link of Π_1. As Π_1 is a proof net, the inner part of $E \lhd F$, which is the trip $F^\uparrow \cdots E^\downarrow$, in $s(\Pi_1)$ contains no conclusion and does not overlap the inner parts of others $\lhd$-link of Π_1. Thus, the unique cycle σ_1 in $s(\Pi_1)$ is $A^\uparrow \cdots F^\uparrow \cdots E^\downarrow \cdots B^\downarrow B^\uparrow \overset{2}{\cdots} A^\downarrow$ or $A^\uparrow \overset{1}{\cdots} B^\downarrow B^\uparrow \cdots F^\uparrow \cdots E^\downarrow \cdots A^\downarrow$.

Let us consider the switching $(s, \lhd_R)$ for Π, the unique cycle σ is $\sigma_1 = A^{\uparrow} \cdots F^{\uparrow} \cdots E^{\downarrow} \cdots B^{\downarrow} A \lhd B^{\downarrow} A \lhd B^{\uparrow} B^{\uparrow} \overset{2}{\cdots} A^{\downarrow}$ or $\sigma_1 = A^{\uparrow} \overset{1}{\cdots} B^{\downarrow} A \lhd B^{\downarrow} A \lhd B^{\uparrow} B^{\uparrow} \cdots F^{\uparrow} \cdots E^{\downarrow} \cdots A^{\downarrow}$. We can easily show that the trip $B^{\uparrow} \cdots A^{\downarrow}$ cannot overlap inner parts of $\lhd$-links of Π_1. A similar argument could be given for the switching $(s, \lhd_L)$. Moreover, by hypothesis the trip $B^{\uparrow} \cdots A^{\downarrow}$ contains no conclusion. Thus, we prove that Π^* is a proof net of MLL and that for every $\lhd_3$-free switching for Π the inner parts of $\lhd$-links in Π contain no conclusion and do not overlap. And finally, Π is a proof net. □

Theorem 4.2 (Termination) *The algorithm always terminates.*

Proof. The main point is to prove that the propagation of the β_i^w-labels always terminates. Let us remind that the propagation and the duplication of β_i^w-labels correspond to the switching positions of the various links (see figure 2). Moreover, from the definition of trips (see [12]), there always exists a trip from $B^{\uparrow}$ to $B^{\downarrow}$. Thus, in the worst case ($B^{\uparrow} \cdots B^{\downarrow}$ contains neither $A^{\downarrow}$ nor $A^{\uparrow}$ nor a conclusion), the propagation of the β_i^w-labels originated from B stops when they move back to B (they stop on the static initial β_i^w-label). Finally, we deduce that the propagation of β_i^w-labels always terminates. □

Theorem 4.3 (Correction) *The algorithm is correct, i.e., if the algorithm builds a proof structure for a given sequent then it is valid.*

Proof. If the algorithm returns a proof net Π for a given sequent, then we can firstly deduce that Π^* is a proof net of MLL, because the algorithm is based on the MLL proof net construction. Moreover, the β_i^w-labels travel throughout the inner parts of the $\lhd$-links. However, if the algorithm succeeds then we can show that no β_i^w-label reaches a conclusion of the proof net. Thus, we can deduce that if the algorithm succeeds then, for every $\lhd_3$-free switching of Π, the inner parts of $\lhd$-links in Π contain no conclusion, which is the condition *ii*) of theorem 4.1. Finally, thanks to the theorem 4.1, we can deduce that Π is a proof net and that the given sequent is valid. □

Theorem 4.4 (Completeness) *The algorithm is complete, i.e., if a given sequent is valid, then the algorithm, applied to this sequent, builds a proof net for it.*

Proof. If S is a valid sequent, then there exists a proof net Π for it. In particular, Π^* is a proof net of MLL. Let us consider a $\lhd$-link, namely $A \lhd B$, in Π. First, as Π^* is a proof net of MLL, the long trips in Π are of the form : $B^{\uparrow} \cdots A^{\downarrow} A^{\uparrow} \cdots B^{\downarrow} A \lhd B^{\downarrow} \cdots A \lhd B^{\uparrow} B^{\uparrow}$ or $B^{\uparrow} \cdots A^{\downarrow} A \lhd B^{\downarrow} \cdots A \lhd B^{\uparrow} A^{\uparrow} \cdots B^{\downarrow} B^{\uparrow}$. Thus, the β_i^w-labels originated from B cannot travel back to B : if they do not stop on a conclusion, they will reach the literal A. Secondly, as for every switching the inner parts of $\lhd$-links contain no conclusion, the β-labels originated from B cannot stop on a conclusion. Thus, the algorithm succeeds and builds a proof net for S. □

5 Conclusion

We have studied two proof-search algorithms for MMLL that are based on proof nets construction and constraints satisfaction. In fact, with the use of labels, the proof-search is guided and controlled from the satisfaction of constraints that are generated during some steps of the proof net construction. Compared to proof-search in sequent calculus, this approach based on proof nets allows to efficiently detect the non-provability [4] and also to prune the proof-search space. The first algorithm (MMLL from NCMLL) is founded on static labels and on a particular satisfaction of constraints that can lead to efficient proof-search, but because of the $\invamp$ associativity it could be necessary in some cases to reconstruct the decomposition tree in order to apply the underlying graph-theoretic characterization for MMLL. The second algorithm (MMLL from MLL) is founded on dynamic labels, that are propagated in the graph structure during the search process, following appropriate rules. In this case there is no global constraints to satisfy like in the former approach but only the compatibility of some labels during the construction of connexions (or axiom-links). Further work could focus on the refinement of the process of constraints satisfaction and also on the generation of countermodels in case of non-provability. Moreover, the algorithms have to be studied more deeply from the algorithmic point of view (data structures or constraints satisfaction). From this work on labelled systems for MMLL, we expect to directly develop new definitions of sequent calculi and proof nets dedicated to the proof-search in MMLL.

References

[1] M. Abrusci and P. Ruet. Non-commutative logic I : the multiplicative fragment. *Annals of Pure and Applied Logic*, 101:29–64, 2000.

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

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

[4] V. Balat and D. Galmiche. *Labelled Deduction*, volume 17 of *Applied Logic Series*, chapter Labelled Proof Systems for Intuitionistic Provability. Kluwer Academic Publishers, 2000.

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

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

[7] G. Delzanno, D. Galmiche, and M. Martelli. A specification logic for concurrent object-oriented programming. *Math. Struct. in Comp. Science*, 9:253–286, 1999.

[8] D. Galmiche. Connection Methods in Linear Logic and Proof nets Construction. *Theoretical Computer Science*, 232(1-2):231–272, 2000.

[9] D. Galmiche and B. Martin. Proof nets construction and automated deduction in non-commutative linear logic - extended abstract. *Electronic Notes in Theoretical Computer Science*, 17, 1998.

[10] 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.

[11] 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.

[12] 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.

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

[14] C. Kreitz and J. Otten. Connection-based theorem proving in classical and non-classical logics. *Journal of Universal Computer Science*, 5(3):88–112, 1999.

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

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

[17] P. Ruet. *Logique non-commutative et programmation par contraintes.* PhD thesis, Université Denis Diderot, Paris 7, 1997.

[18] P. Ruet and F. Fages. Concurrent constraint programming and non-commutative logic. In *11th Int. Workshop on Computer Science Logic, CSL'97, LNCS 1414*, pages 406–423, Aarhus, Denmark, August 1997.

[19] 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.

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

A Rules for β_i^w-labels propagation

In this section we present all the propagation rules for the β_i^w-labels, which can be applied during the proof-search algorithm presented in section 4.2.

Rules for axiom-links

The propagation rules for axiom-links are simple : when a β_i^w-label moves upward to A (resp. $A^\perp$), it then moves downward from $A^\perp$ (resp. A).

Fig. A.1. Rules for the axiom-links

Rules for $\odot$-links

The rules for the $\odot$-links are quite simple when compared to the ones for $\otimes$, $⅋$ and $\lhd$ links. In fact, when a β_i^w-label moves through a $\odot$-link, it does not duplicate. This is related to the fact that a $\odot$ link has only one switching position.

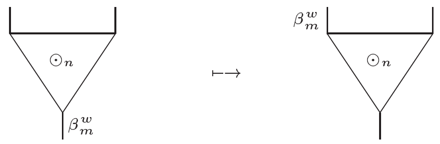


Fig. A.2. Upward rules for $\odot$-links

The reader must be aware of the fact that the direction of propagation of the label may change. For example, when a β_i^w moves downward to the left premise of a $\odot$-link, it then moves upward from the right premise of the link. At the opposite, if a β_i^w label moves upward to the conclusion of a $\odot$-link, it then moves upward from the left premise of the link.

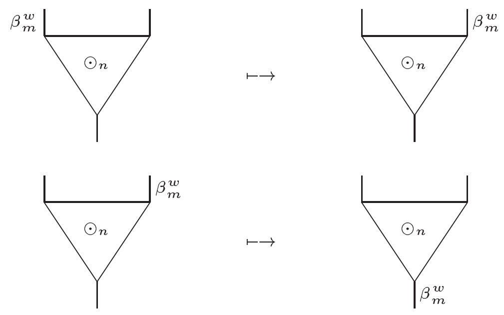


Fig. A.3. Downward rules for $\odot$-links

Rules for $\otimes$-links

The propagation rules for the $\otimes$-links are more complicated than the ones for $\odot$-links. In fact, the propagation has to take into account that in the initial definition of links the $\otimes$-links have two switching positions. This explains why β_i^w-labels have to be duplicated and also the way to use the words w to guide the traversal of the structure.

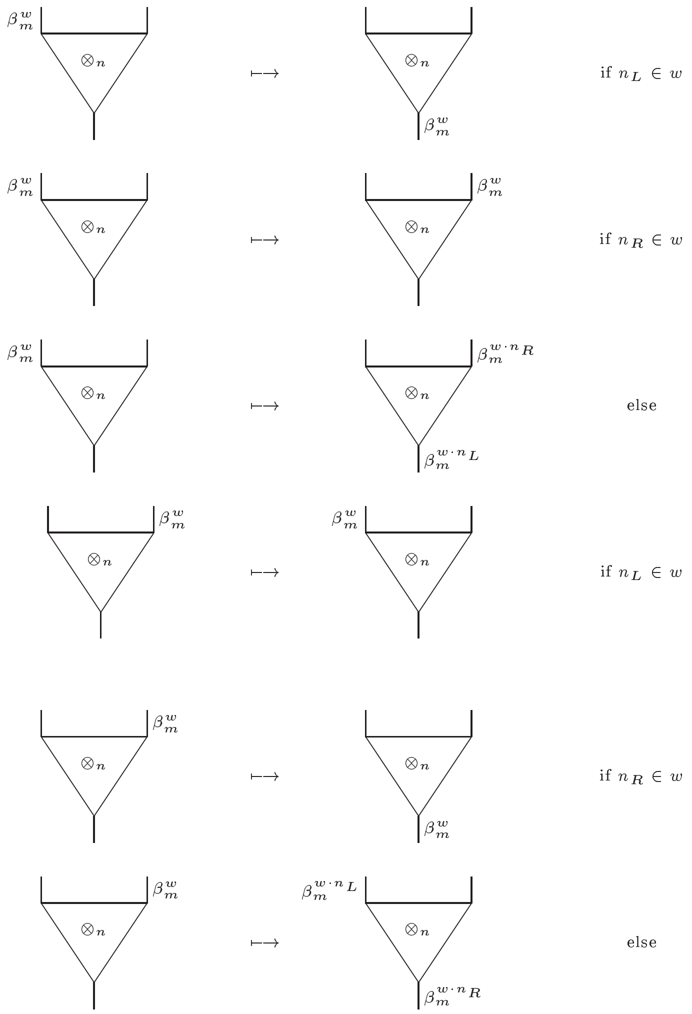


Fig. A.4. Downward rules for $\otimes$-links

Rules for $\parr$-links

As for the $\otimes$-links, a β_i^w-label that moves through a $\parr$-link may be duplicated. The direction of propagation may change during the moves through a $\parr$-link. For example, if we consider a $\parr$-link numbered j, and a β_i^w-label where w contains j_L, if it moves downward to the right premise of the link, then it moves upward from the same premise.

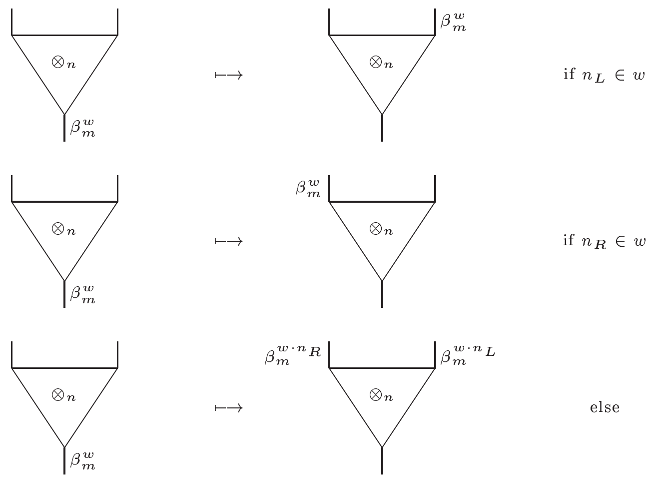


Fig. A.5. Upward rules for $\otimes$

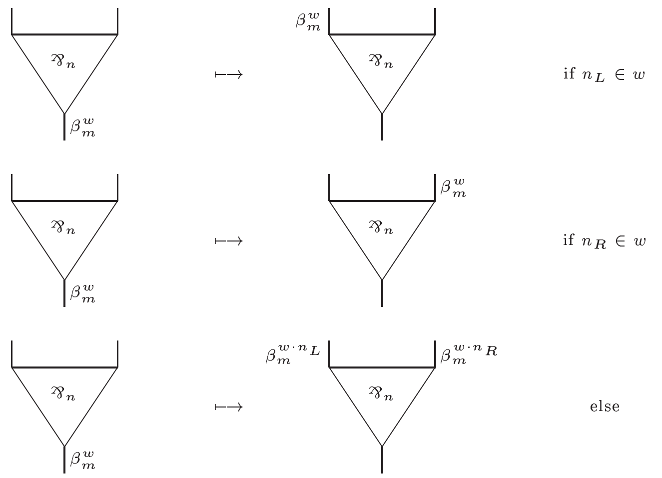


Fig. A.6. Upward rules for $\parr$-links

Rules for $\lhd$-links

The propagation rules for $\lhd$-links when $i \neq j$, are the same as for the $\parr$-links. In the case $i = j$, the β_i^w-label does not cross the link, it stops accordingly to the rules defined in section 4.2.

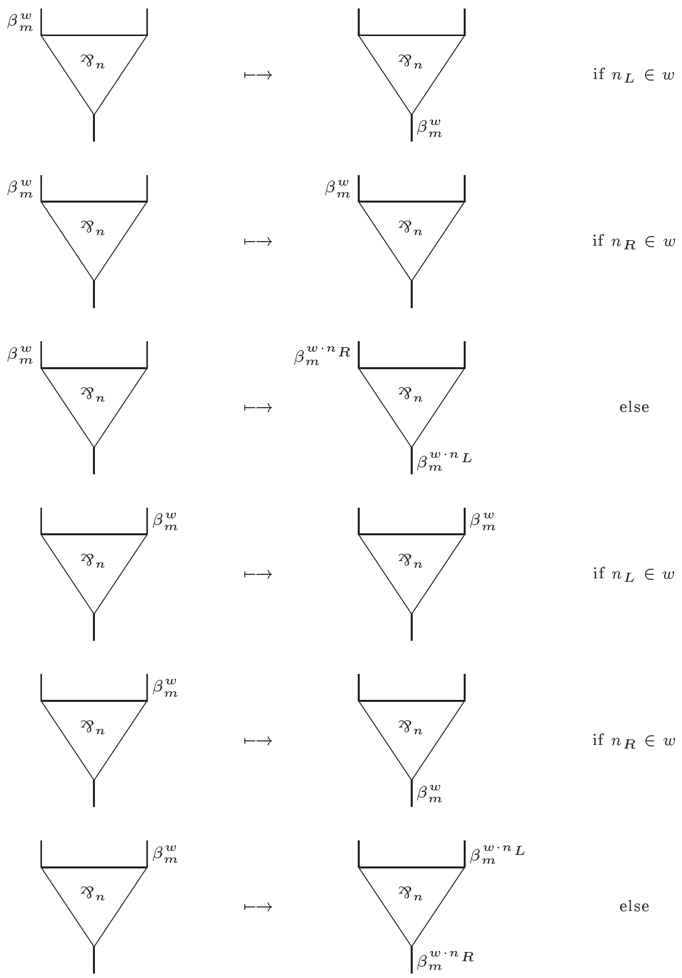


Fig. A.7. Downward rules for ⅋-links

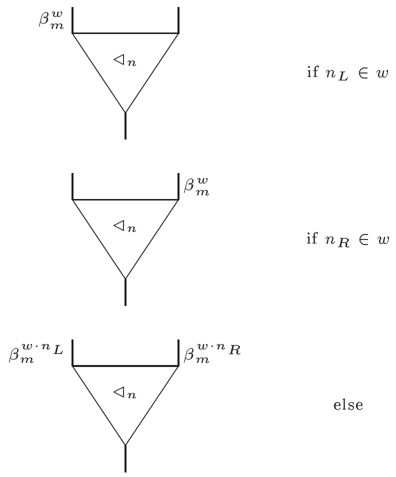


Fig. A.8. Upward rules for $\lhd$-links

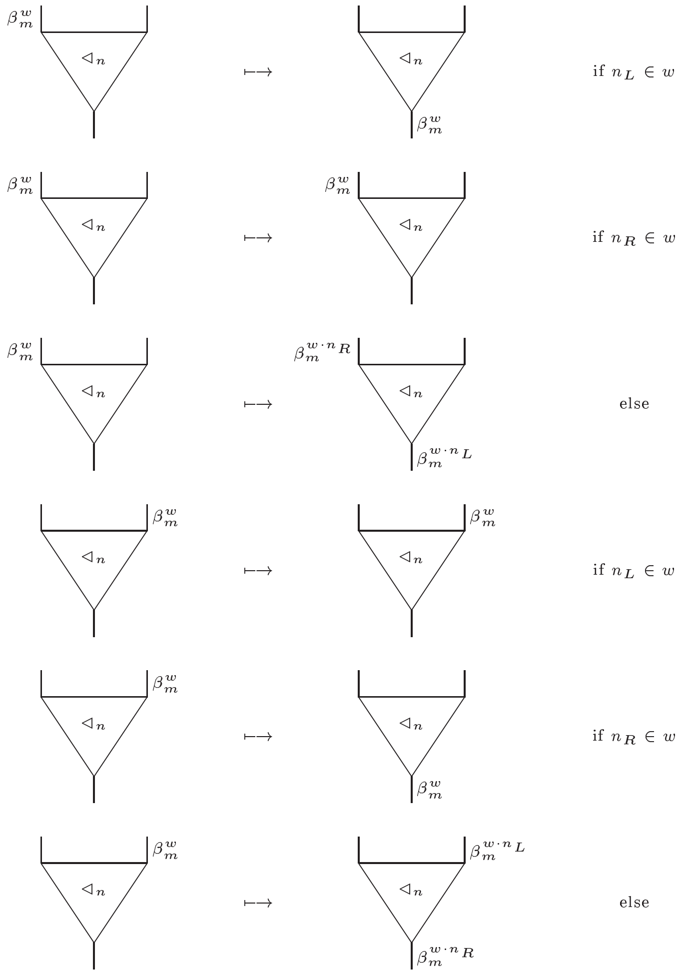


Fig. A.9. Downward rules for $\lhd$-links