Calculi with dependency relations for Mixed Linear Logic - extended abstract -

D. Galmiche, J.M. Notin

LORIA–Université Henri Poincaré
Campus Scientifique, BP 239
Vandœuvre-lès-Nancy, France

Abstract

In this paper we propose new calculi for the multiplicative fragment of Mixed Linear Logic (MMLL) which is a logic that combines both commutative and non-commutative connectives. These based-on sequent and proof net calculi, that can be seen as a new proof-theoretical formulation of MMLL, are based on the definition of dependency relations. We provide a proof-search procedure for MMLL that is based on proof nets construction with associated sets of dependencies.

1 Introduction

Many works have been recently devoted to the study of linear logic [10] (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 the use of non-commutative logic [2] or non-commutative connectives to deal with sequentiality. Recently, a mixed version of classical propositional linear logic, which combines both commutative and non-commutative connectives, has been proposed [1] with the motivation to define logical characterizations of synchronization mechanisms in concurrent constraint programming [15]. But in order to apply such a logic to process calculi and concurrent programming (specification, verification) it is necessary to have new proof systems and related efficient proof-search methods.

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). We naturally consider our previous works, based on the concept of proof net, dedicated to proof search in both fragments [6,7] but in MMLL an essential step consists in understanding and analyzing the interaction between commutative and non-commutative connectives during the proof-search process. We have recently studied two proof search methods in MMLL that were based on the definition of labelled (with constraints) proof nets [8], the labels allowing in the former to fix some constraints to satisfy during the proof net construction 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.

In this paper, we propose new sequent and proof net calculi for the multiplicative fragment of Mixed Linear Logic (MMLL). These calculi are based on the definition of dependency-relations and provide a new formulation of MMLL based on such relations in the spirit of [5]. Such new proof systems allow to naturally consider this logical fragment as a particular refinement of the MLL fragment. Moreover the definition of dependencies (or constraints) attached to the calculi is a natural way to translate some proof-theoretical results of MMLL into simple and tractable automated proof-search procedures, that are based on the construction of semantical structures like proof nets [6,11]. After the study of the relationships between both systems (adequacy and sequentialization), we propose a based-on proof nets algorithm for proof construction in MMLL that builds in parallel a proof structure (like in MLL) and a set of dependencies on which we have to verify some validity conditions. In order to have a simple and efficient verification process on the dependencies, we propose refinements of both calculi that include dependencies with constraints. Such systems will be more deeply studied in order to analyze the execution of programs or to prove systems properties specified in such a mixed logic. Moreover our new proof net definition provides a new correctness criterion for MMLL that will be studied from both complexity and algorithmic points of view.

2 Mixed Non-Commutative 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 [15]. This logic is an alternative to commutative classical, intuitionistic or linear logics for applications in

[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.

linguistics or computer science (process calculi or concurrent constraint programming [15]).

2.1 Motivations and applications

Let us first illustrate the motivations and the interest to have a logical fragment where commutative and non-commutative connectives cohabit, with an example coming from the concurrent constraint (CC) programming [15].
Let us consider the problem of two processes (p_1 and p_2) that request the same two resources (r_1 and r_2). It can be expressed in CC by

$$P = tell(r_1) \parallel tell(r_2) \parallel \underbrace{r_1 \to r_2 \to \big(tell(r_1) \parallel tell(r_2)\big)}_{p_1} \parallel \underbrace{r_2 \to r_1 \to \big(tell(r_2) \parallel tell(r_1)\big)}_{p_2}$$

In this expression $\parallel$ represents the parallel composition of processes ; $tell(r_1)$ allows to state that the resource r_1 is available. Moreover $p_1 \equiv r_1 \to p_1'$ expresses that p_1 waits for the resource r_1 to be available, and then consumes this resource to become p_1'.
Obviously, this situation can lead to a deadlock, that is to say :

$$D = \underbrace{r_2 \to \big(tell(r_1) \parallel tell(r_2)\big)}_{p_1'} \parallel \underbrace{r_1 \to \big(tell(r_2) \parallel tell(r_1)\big)}_{p_2'}$$

where p_1' and p_2' both wait for resources that are no more available, because p_1 has consumed the resource r_1, and p_2 has consumed the resource r_2.

In order to characterize such properties, programs or expressions like P or D can be translated into a particular logic. For instance, P is translated into commutative linear logic, in which formulae are considered as resources, as follows:

$$\hat{P} = r_1 \otimes r_2 \otimes (r_1 \multimap (r_2 \multimap (r_1 \otimes r_2))) \otimes (r_2 \multimap (r_1 \multimap (r_1 \otimes r_2)))$$

where $\multimap$ is the linear implication, and $\otimes$ is the multiplicative conjonction.

Unfortunately, because of logical interactions between $\otimes$ and $\multimap$, we cannot use the validity in commutative linear logic in order to state that the program P can lead to the above deadlock D.
It appears necessary to use a non-commutative linear implication $-\!\bullet$, together with $\otimes$, in order to express that the program P can lead to the deadlock situation D. The formula representing our program is :

$$\check{P} = r_1 \otimes r_2 \otimes (r_1 -\!\bullet(\Diamond \otimes (r_2 -\!\bullet(\Diamond \otimes r_1 \otimes r_2)))) \otimes (r_2 -\!\bullet(\Diamond \otimes (r_1 -\!\bullet(\Diamond \otimes r_2 \otimes r_1))))$$

where $\Diamond$ is a new atomic formula.
Following the same translation, the deadlock D is represented by the formula :

$$\big(r_2 -\!\bullet(\Diamond \otimes r_1 \otimes r_2)\big) \otimes \big(r_1 -\!\bullet(\Diamond \otimes r_1 \otimes r_2)\big)$$

Thus the fact that P can lead to D is ensured by the validity of the sequent $r_1 \otimes r_2 \otimes (r_1 \multimap\!\bullet (\diamond \otimes p'_1)) \otimes (r_2 \multimap\!\bullet (\diamond \otimes p'_2)) \vdash \diamond \otimes \diamond \otimes 1 \otimes p'_1 \otimes p'_2$ (where $p'_1 = r_2 \multimap\!\bullet (\diamond \otimes r_1 \otimes r_2)$ and $p'_2 = r_1 \multimap\!\bullet (\diamond \otimes r_1 \otimes r_2)$) in Mixed Linear Logic. It can be done by proof-search for this sequent in one of the sequent calculi of this logic.

Therefore, we see that a logic like linear logic, in which formulae can be seen and treated as resources or agents, is necessary to some specificy properties of real systems. Moreover, the use of mixed linear logic, where commutative and non-commutative connectives cohabit, is necessary to obtain the logical characterization of finer operational aspects of concurrent constraints computations (stores, successes and suspensions) with synchronization constraints [15].

In addition to this ability to represent mechanisms or properties dealing with sequentiality and concurrency, this logic can also offer new perspectives to the connection between concurrency and proof theory, with the paradigm of logic programming (proof-search as computation). Therefore, one could propose new definitions or extensions of process calculi [4,12] where sequentiality and concurrency can cohabit, following such a correspondence with Mixed Linear Logic. It suggests new investigations of the semantics of concurrency from proof theory.

In order to study the execution of programs (agents) or to prove systems properties specified in such a mixed logic, appropriate proof systems (sequent or proof net calculi) are required. We will propose such new calculi in this paper, focusing on the syntax of proof nets (particular proof graphs in linear logic) that is more suited for the representation of concurrent computations [11] but, as recently shown, also for proof construction in commutative and non-commutative linear logic [7].

2.2 A sequent calculus with explicit structural rules

The formulae (of MMLL) are build from literals $p, q, \ldots, p^\perp, q^\perp, \ldots$ with non-commutative connectives (conjunction $\odot$ and disjunction $\lhd$), commutative connectives (conjunction $\otimes$ and disjunction $\parr$). The negation is defined as follows with the De Morgan laws: $(p)^\perp = p^\perp$, $(p^\perp)^\perp = p$, $(A \odot B)^\perp = B^\perp \lhd A^\perp$, $(A \lhd B)^\perp = B^\perp \odot A^\perp$, $(A \otimes B)^\perp = B^\perp \parr A^\perp$, $(A \parr B)^\perp = B^\perp \otimes A^\perp$. The formulae of MLL (resp. NCMLL) are build from the atoms $p, q, \ldots, p^\perp, q^\perp, \ldots$ and the connectives $\otimes$, $\parr$, (resp. $\odot$ and $\lhd$).
There exists a sequent calculus with explicit structural rules for MMLL [1]. To mix commutative and non-commutative connectives in a same sequent calculus, two separators of formulae are used in a sequent: a commutative one

$$\frac{}{\vdash A^{\perp}, A}\,ax \qquad \frac{\vdash \Gamma, A \qquad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}\,cut \qquad \frac{\vdash \Gamma[\Delta;\Sigma]}{\Gamma[\Delta,\Sigma]}\,entr$$

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

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

Fig. 1. A sequent calculus for MMLL

',' and a non-commutative one ';'. In this MMLL sequent calculus (see figure 1), some explicit structural rules (*seesaw* and *entropy*) define the relationships between these separators. Moreover they are used in conjunction with the rule for the commutativity of ',' in order to recover the cyclicity of ';'. 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 proof-theoretic and semantic results about MMLL (cut-elimination, sub-formula property, phase semantics, completeness theorem) are given in [1,14]. If we aim to analyze the proof search with this calculus, some problems arise: some orders are fixed by the non-commutative connectives and the commutative and non-commutative connectives interact in the same framework. Let us note that an alternative to study the proof theory of MMLL is to consider the notion of proof net [11].

2.3 A Sequent calculus with implicit structural rules

In [1], P. Ruet introduced a sequent calculus which used a specific mathematical structure, namely order variety, to deal with the interaction between commutative and non-commutative connectors.

2.3.1 Order varieties

We recall here the main definitions and theorems about order varieties [1,14]. Order varieties can be seen as refinements of cyclic orders and are structures which, provided a point of view (an element x in the base set), can be seen as partial orders on the complement of $\{x\}$.

Definition 1 (order variety) *Let E be a set, an* order variety *on E is a ternary relation α which is :*
- *cyclic :* $\forall x, y, z \in E\ \alpha(x, y, z) \Rightarrow \alpha(y, z, x)$
- *anti-reflexive :* $\forall x, y \in E\ \neg\alpha(x, x, y)$
- *transitive :* $\forall x, y, z, t \in E\ \alpha(x, y, z) \wedge \alpha(z, t, x) \Rightarrow \alpha(y, z, t)$
- *spreading :* $\forall x, y, z, t \in E\ \alpha(x, y, z) \Rightarrow \alpha(t, y, z) \vee \alpha(x, t, z) \vee \alpha(x, y, t)$.

An order variety is said total *when :* $\forall x, y, z \in E\ x \neq y \neq z \Rightarrow \alpha(x,y,z) \vee \alpha(z,y,x)$

Any order variety α on a set E induces an oriented graph $\rightarrow_\alpha$ on E with an edge between x and $y \in E$ iff : $\forall z \in E\ z \neq x \wedge z \neq y \Rightarrow \alpha(x,y,z)$. If α is a total order variety, then $\rightarrow_\alpha$ is an oriented cycle.
Let α be an order variety on E and $x \in E$, the binary relation α_x on $E \setminus \{x\}$ is defined by $\alpha_x(y,z)$ iff $\alpha(x,y,z)$.

Definition 2 *Let (E,ω) be a strict order and $z \in E$, the binary relation $\overset{z}{<}_\omega$ is defined by :*
$x \overset{z}{<}_\omega y$ iff $x <_\omega y$ and z is comparable with neither x nor y.
The ternary relation $\overline{\omega}$ on E is defined by:

$$\overline{\omega}(x,y,z)\ iff\ \begin{cases} x <_\omega y <_\omega z\ \vee\ z <_\omega x <_\omega y\ \vee\ y <_\omega z <_\omega x\ \vee \\ x \overset{z}{<}_\omega y\ \vee\ y \overset{x}{<}_\omega z\ \vee\ z \overset{y}{<}_\omega x \end{cases}$$

Definition 3 *Let ω_1 and ω_2 be orders on disjoints sets X and Y. We define their parallel ($\omega_1 \parallel \omega_2$) and serial ($\omega_1 \ll \omega_2$) composition, that are both orders on $X \cup Y$, by :*
- $(\omega_1 \parallel \omega_2)(x,y)$ iff $x <_{\omega_1} y$ or $x <_{\omega_2} y$,
- $(\omega_1 \ll \omega_2)(x,y)$ iff $x <_{\omega_1} y$ or $x <_{\omega_2} y$ or $(x \in X$ and $y \in Y)$.

Definition 4 (interior) *Let α be an order variety on E, its interior is defined by: $\natural\alpha = \bigcap_{x \in E} \overline{\alpha_x \parallel x}$*

2.3.2 A sequent calculus with order varieties

Informally, the inference rules are those of the sequent calculus for MLL, seeing $\triangleleft$ as a $⅋$, and $\odot$ as a $\otimes$. In order to recover the non-commutative behavior of $\triangleleft$ and $\odot$, sequents are associated with order varieties. Thus, when an inference rule is applied, the order variety corresponding to the conclusion of the rule is obtained from the order varieties associated to the premisse. Moreover, the rule for $\triangleleft$ can be applied only if a particular condition about the related order variety is satisfied.

Definition 5 (Sequents) *A sequent of MMLL $\vdash \Gamma\ \langle\alpha\rangle$ is a multiset Γ of occurrences of formulae, together with an order variety α on Γ.*

The rules of the sequent calculus are given in figure 2. Let us note that they use operations on order varieties defined as follows: let α and β be two order varieties respectively defined on $\Gamma \cup \{A\}$ and $\Delta \cup \{B\}$, $\alpha \odot_{A,B} \beta$ is the order variety on $\Gamma \cup \Delta \cup \{A \odot B\}$ defined by $\alpha \odot_{A,B} \beta = \overline{\alpha_A \ll A \odot B \ll \beta_B}$. Moreover $\alpha \otimes_{A,B} \beta$ is the order variety on $\Gamma \cup \Delta \cup \{A \otimes B\}$ defined by $\alpha \otimes_{A,B} \beta =$

$$\frac{}{\vdash A^\perp, A\ \langle\emptyset\rangle}\ ax \qquad \frac{\vdash \Gamma, A\ \langle\alpha\rangle \qquad \vdash B, \Delta\langle\beta\rangle}{\vdash \Gamma, A \odot B, \Delta\ \langle\alpha \odot_{A,B} \beta\rangle}\ \odot$$

$$\frac{\vdash \Gamma, A, B\ \langle\alpha\rangle}{\vdash \Gamma, A \lhd B\ \langle\alpha[A \lhd B/A,B]\rangle}\ \lhd \text{ if } A \rightarrow_\alpha B$$

$$\frac{\vdash \Gamma, A\ \langle\alpha\rangle \qquad \vdash B, \Delta\ \langle\beta\rangle}{\vdash \Gamma, A \otimes B, \Delta\ \langle\alpha \otimes_{A,B} \beta\rangle}\ \otimes \qquad \frac{\vdash \Gamma, A, B\ \langle\alpha\rangle}{\vdash \Gamma, A \parr B\ \langle\alpha[A \parr B/A,B]\rangle}\ \parr$$

Fig. 2. A sequent calculus with order varieties

$\overline{\alpha_A \parallel A \otimes B \parallel \beta_B}$. In addition, let α be an order variety on a set $E \cup \{x,y\}$ with $x, y \notin E$, and let $z \in E$, we define the pasting $\alpha[z/x,y]$ of x and y along z in α by : $\alpha[z/x,y] = \natural\left(\alpha\restriction_{E\cup\{x\}}[z/x] \bigcap \alpha\restriction_{E\cup\{y\}}[z/y]\right)$. It is important to notice that the $\lhd$ rule can only be applied under the condition $A \rightarrow_\alpha B$. This point is essential to understand the contributions of the next section.

Let us give a proof of the sequent $\vdash (A \odot B) \lhd C, A^\perp, C^\perp \odot B^\perp$.

$$\dfrac{\dfrac{\dfrac{}{\vdash C^\perp, C\ \langle\emptyset\rangle}\,(ax) \quad \dfrac{\dfrac{}{\vdash A^\perp, A\ \langle\emptyset\rangle}\,(ax) \quad \dfrac{}{\vdash B^\perp, B\ \langle\emptyset\rangle}\,(ax)}{\vdash A^\perp, A \odot B, B^\perp\ \langle\overline{A^\perp < A \odot B < B^\perp}\rangle}\,(\odot)}{\vdash C, C^\perp \odot B^\perp, A^\perp, A \odot B\ \langle\overline{C < C^\perp \odot B^\perp < A^\perp < A \odot B}\rangle}\,(\odot)}{\vdash (A \odot B) \lhd C, A^\perp, C^\perp \odot B^\perp\ \langle\overline{C^\perp \odot B^\perp < A^\perp < (A \odot B) \lhd C}\rangle}\,(\lhd)$$

This derivation is a proof of the sequent because the condition to apply the $\lhd$ rule is satisfied. We observe that this verification cannot be trivially included in the process of proof search within this calculus. That is one of the main interest of a new calculus with dependencies in which such a condition is more naturally expressed inside the calculus itself.

2.4 Proof search in MMLL

The calculus with explicit structural rules is clearly not appropriate for automated proof search because of the difficult management of the structural rules during the proof search process. In the opposite, the calculus with implicit structural rules has no more two formulae separators to deal with, and the underlying sequent calculus is in fact the MLL sequent calculus, for which we have results on proof normalization [9] and proof-search [6]. But the order varieties, associated to sequents, are mathematical structures which are difficult to deal with during proof-search.

We have recently study proof-search in MMLL from both commutative and non-commutative linear logics and thus have defined two different proof-search algorithms, both based on the construction of labelled proof nets [8]. 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. Moreover, such based-on proof nets methods are useful in order to efficiently detect the non-provability.

Our aim is thus to propose a new formulation of MMLL based on dependency-relations with a new sequent calculus but also a new proof net definition that is another view to present the interactions between commutative and non-commutative connectives in the same logic. Moreover, the based-on proof nets calculus we propose is more appropriate to deal with proof-search in MMLL following our based-on proof nets approach of proof construction in commutative and non-commutative MLL [6,7]. Comparisons with our previous proposals [8] will be studied.

Our aim here is proof construction but we will see that our algorithm, based on proof net construction, can also be adapted and used for proof verification. Our new proof net definition provides a new correctness criterion for MMLL and thus our algorithm provides a new way to verify correctness. It will be compared to the recent quadratic correctness criterion proposed in [13].

3 A sequent calculus with dependency relations

We now introduce a new sequent calculus, which aims to be more adapted for proof search and proof construction in MMLL. This calculus use the notion of *dependency* in order to express some order between formulae.

Definition 6 *A* dependency relation *between two formulae A and B is represented by $A \to B$. A sequent with dependencies $\vdash \Gamma\ \mathcal{D}$ is a multiset Γ of formulae, associated with a set of dependencies $\mathcal{D}$.*

Let γ be a subset of Γ. Then, $\mathcal{D}\restriction_\gamma$ denotes the restriction of $\mathcal{D}$ to the subset γ of formulae. Thus we have $\mathcal{D}\restriction_\gamma = \{E \to F | E, F \in \gamma, E \to F \in \mathcal{D}\}$. Moreover $\mathcal{D}[Y/X]$ represents the set $\mathcal{D}$ where the formula (occurrences) X is replaced by Y.

A set of dependencies $\mathcal{D}$ is a graph the nodes of which are the formulae of Γ and with edges representing dependency relations between formulae. By construction, a set of dependencies is cyclic.

Definition 7 *A cycle is said* maximal *in $\mathcal{D}$ if it exactly contains once all the formulae of Γ.*

By construction, there always exists a maximal cycle in a set $\mathcal{D}$.
Let $\vdash \Gamma\ \mathcal{D}$ be a sequent with dependencies, the set of maximal cycles in $\mathcal{D}$ is denoted $\mathcal{D}_C$.

3.1 A new sequent calculus

We give a sequent calculus with dependencies in figure 3. In fact, we observe that the sequent rules are structurally the same for $\otimes$ and $\odot$ and also for $\parr$ and $\lhd$ and that the semantical differences are in fact in the construction of the dependency relations. Let us note that the rules $\frac{\vdash A\ \mathcal{D}_1 \qquad \vdash B,\Delta\ \mathcal{D}_2}{\vdash A\odot B,\Delta\ \mathcal{D}}\odot$ and $\frac{\vdash \Gamma,A\ \mathcal{D}_1 \qquad \vdash B\ \mathcal{D}_2}{\vdash \Gamma,A\odot B\ \mathcal{D}}\odot$ are special cases of the sequent rule $\frac{\vdash \Gamma,A\ \mathcal{D}_1 \qquad \vdash B,\Delta\ \mathcal{D}_2}{\vdash \Gamma,A\odot B,\Delta\ \mathcal{D}}\odot$. The distinction between these cases allows to simplify the calculus of the dependencies. A similar remark can be made for the $\otimes$-rule.
Moreover, looking at the rule for $\otimes$, we can see that the dependencies are extended as for $A \odot B$, and also as for $B \odot A$. This difference in the construction of the dependencies from those associated to the premisse allows, in some way, to express the commutativity of $\otimes$. Actually, the difference is made essentially on the maximal cycles, especially when checking the condition on the application of the $\lhd$-rule.

Before to present the properties of this calculus, we give an example of derivation with the sequent $\vdash (A \otimes B) \lhd C, A^\perp, C^\perp \odot B^\perp$. A proof is the classical sequent tree where a set of dependencies is associated to each sequent. The important point is that a rule application modifies the set of dependencies $\mathcal{D}_i$ and that when one applies a $\lhd$ rule we have to verify the condition about the membership to the maximal cycles.

$$
\cfrac{\cfrac{\cfrac{}{\vdash C^\perp, C\ \mathcal{D}_3}(Ax) \qquad \cfrac{\cfrac{}{\vdash A^\perp, A\ \mathcal{D}_1}(Ax) \qquad \cfrac{}{\vdash B^\perp, B\ \mathcal{D}_2}(Ax)}{\vdash A^\perp, A\otimes B, B^\perp\ \mathcal{D}_4}(\otimes)}{\vdash C, C^\perp\odot B^\perp, A^\perp, A\otimes B\ \mathcal{D}_5}(\odot)}{\vdash (A\otimes B)\lhd C, A^\perp, C^\perp\odot B^\perp\ \mathcal{D}_6}(\lhd)
$$

where $\mathcal{D}_1 = \left\{ \begin{array}{c} A \to A^\perp \\ A^\perp \to A \end{array} \right\}$, $\mathcal{D}_2 = \left\{ \begin{array}{c} B \to B^\perp \\ B^\perp \to B \end{array} \right\}$, $\mathcal{D}_3 = \left\{ \begin{array}{c} C \to C^\perp \\ C^\perp \to C \end{array} \right\}$.

We have $\mathcal{D}_4 = \left\{ \begin{array}{cc} B^\perp \to A\otimes B & B^\perp \to A^\perp \\ A\otimes B \to A^\perp & A^\perp \to A\otimes B \\ A^\perp \to B^\perp & A\otimes B \to B^\perp \end{array} \right\}$,

and finally $\mathcal{D}_5 = \left\{ \begin{array}{cc} C \to A\otimes B & C \to A^\perp \\ A\otimes B \to A^\perp & A^\perp \to A\otimes B \\ A^\perp \to C^\perp\odot B^\perp & A\otimes B \to C^\perp \odot B^\perp \\ C^\perp\odot B^\perp \to C & \end{array} \right\}$.

$$\frac{}{\vdash A^{\perp}, A \;\; \mathcal{D}} Ax \qquad \text{with } \mathcal{D} = \left\{ \begin{array}{ccc} A & \rightarrow & A^{\perp} \\ A^{\perp} & \rightarrow & A \end{array} \right\}$$

$$\frac{\vdash \Gamma, A \;\; \mathcal{D}_1 \qquad \vdash B, \Delta \;\; \mathcal{D}_2}{\vdash \Gamma, A \odot B, \Delta \;\; \mathcal{D}} \odot \qquad \text{with} \quad \begin{array}{rcl} \mathcal{D} & = & \mathcal{D}_1 \upharpoonright_{\Gamma} \cup \mathcal{D}_2 \upharpoonright_{\Delta} \cup \\ & & \{E \rightarrow F | E \in \Gamma, F \in \Delta, E \rightarrow A \in \mathcal{D}_1, \\ & & B \rightarrow F \in \mathcal{D}_2\} \cup \\ & & \{A \odot B \rightarrow E | E \in \Gamma, A \rightarrow E \in \mathcal{D}_1\} \cup \\ & & \{F \rightarrow A \odot B | F \in \Delta, F \rightarrow B \in \mathcal{D}_2\} \end{array}$$

$$\frac{\vdash A \;\; \mathcal{D}_1 \qquad \vdash B, \Delta \;\; \mathcal{D}_2}{\vdash A \odot B, \Delta \;\; \mathcal{D}} \odot \qquad \text{with } \mathcal{D} = \mathcal{D}_2[A \odot B / B]$$

$$\frac{\vdash \Gamma, A \;\; \mathcal{D}_1 \qquad \vdash B \;\; \mathcal{D}_2}{\vdash \Gamma, A \odot B \;\; \mathcal{D}} \odot \qquad \text{with } \mathcal{D} = \mathcal{D}_1[A \odot B / A]$$

$$\frac{\vdash \Gamma, A, B \;\; \mathcal{D}_1}{\vdash \Gamma, A \lhd B \;\; \mathcal{D}} \lhd \qquad \begin{array}{l} \text{iff } B \rightarrow A \text{ is in all the maximal cycles in } \mathcal{D}_1 \\ \text{with } \mathcal{D} = \mathcal{D}_1[A \lhd B / A, A \lhd B / B] \end{array}$$

$$\frac{\vdash \Gamma, A \;\; \mathcal{D}_1 \qquad \vdash B, \Delta \;\; \mathcal{D}_2}{\vdash \Gamma, A \otimes B, \Delta \;\; \mathcal{D}} \otimes \qquad \text{with} \quad \begin{array}{rcl} \mathcal{D} & = & \mathcal{D}_1 \upharpoonright_{\Gamma} \cup \mathcal{D}_2 \upharpoonright_{\Delta} \cup \\ & & \{E \rightarrow F | E \in \Gamma, F \in \Delta, E \rightarrow A \in \mathcal{D}_1, \\ & & B \rightarrow F \in \mathcal{D}_2\} \cup \\ & & \{A \otimes B \rightarrow E | E \in \Gamma, A \rightarrow E \in \mathcal{D}_1\} \cup \\ & & \{F \rightarrow A \otimes B | F \in \Delta, F \rightarrow B \in \mathcal{D}_2\} \cup \\ & & \{F \rightarrow E | E \in \Gamma, F \in \Delta, A \rightarrow E \in \mathcal{D}_1, \\ & & F \rightarrow B \in \mathcal{D}_2\} \cup \\ & & \{A \otimes B \rightarrow F | F \in \Delta, B \rightarrow F \in \mathcal{D}_2\} \cup \\ & & \{E \rightarrow A \otimes B | E \in \Gamma, E \rightarrow A \in \mathcal{D}_1\} \end{array}$$

$$\frac{\vdash A \;\; \mathcal{D}_1 \qquad \vdash B, \Delta \;\; \mathcal{D}_2}{\vdash A \otimes B, \Delta \;\; \mathcal{D}} \otimes \qquad \text{with } \mathcal{D} = \mathcal{D}_2[A \otimes B / B]$$

$$\frac{\vdash \Gamma, A \;\; \mathcal{D}_1 \qquad \vdash B \;\; \mathcal{D}_2}{\vdash \Gamma, A \otimes B \;\; \mathcal{D}} \otimes \qquad \text{with } \mathcal{D} = \mathcal{D}_1[A \otimes B / A]$$

$$\frac{\vdash \Gamma, A, B \;\; \mathcal{D}_1}{\vdash \Gamma, A \parr B \;\; \mathcal{D}} \parr \qquad \text{with} \quad \begin{array}{rcl} \mathcal{D} & = & \mathcal{D}_1[A \parr B / A, A \parr B / B] \cup \\ & & \{E \rightarrow F | E, F \in \Gamma, \; E \rightarrow A, A \rightarrow F \in \mathcal{D}_1\} \cup \\ & & \{E \rightarrow F | E, F \in \Gamma, \; E \rightarrow B, B \rightarrow F \in \mathcal{D}_1\} \end{array}$$

Fig. 3. A sequent calculus with dependencies

There are two maximal cycles in $\mathcal{D}_5$, namely $C \rightarrow A \otimes B \rightarrow A^{\perp} \rightarrow C^{\perp} \odot B^{\perp} \rightarrow C$ and $C \rightarrow A^{\perp} \rightarrow A \otimes B \rightarrow C^{\perp} \odot B^{\perp} \rightarrow C$. Then we can deduce that $C \rightarrow A \otimes B$ does not belong to the maximal cycles of $\mathcal{D}_5$. Then, we cannot apply any rule to obtain $(A \otimes B) \lhd C$ and therefore the given sequent is not provable.

If we replace $(A \otimes B)$ by $(A \odot B)$ we have a similar proof tree but with another set $\mathcal{D}_5$. In this case, the only maximal cycle in $\mathcal{D}_5$ is $C \to A \odot B \to A^\perp \to C^\perp \odot B^\perp \to C$ and therefore $C \to A \odot B$ is in all the maximal cycles in $\mathcal{D}_5$. Thus, the given sequent $\vdash (A \odot B) \lhd C, A^\perp, C^\perp \odot B^\perp$ is provable.

The main point for the proof construction within this calculi is the construction of the set of dependencies associated to the sequents and the verification for the $\lhd$ rules of the condition for a special dependency to belong to all maximal cycles of the related dependencies set.
In the next section, we propose a new calculus based on proof nets with dependencies and an algorithm for the construction of such proof nets in order to analyze the provability in MMLL. Before we prove the equivalence between this new calculus and the initial calculus with order varieties.

3.2 Properties of the calculus

Theorem 8 *Let Π be a proof in the sequent calculus with dependencies, with conclusion $\vdash \Gamma\ \mathcal{D}$. Then, there exists a maximal cycle in $\mathcal{D}$.*

PROOF. By induction on the structure of Π. $\square$

Theorem 9 *The sequent calculus with dependencies is equivalent to the sequent calculus with order varieties.*

PROOF. The main point is that when deleting the structure associated to sequents (order varieties or dependencies), the calculus is the one for MLL, considering the connectives $\odot$ and $\lhd$ as respectively $\otimes$ and $\parr$. Let us consider the $\lhd$-rule in both sequent calculus :

$$\frac{\vdash \Gamma, A, B\ \langle\alpha\rangle}{\vdash \Gamma, A \lhd B\ \langle\alpha[A \lhd B/A, B]\rangle}\lhd \qquad \frac{\vdash \Gamma, A, B\ \mathcal{D}_1}{\vdash \Gamma, A \lhd B\ \mathcal{D}}\lhd$$

Thus, to prove the equivalence of both calculus, we only need to prove the equivalence of the conditions of the $\lhd$-rule, that is to say :

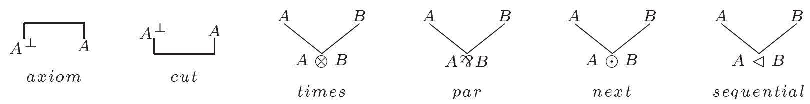
$$A \to_\alpha B \text{ iff } B \to A \text{ is in all the maximal cycles of } \mathcal{D}_1$$

This can be shown directly by induction, using the results on order varieties. The proof is not detailed here, but it will follow from the results on the use of dependencies for proof nets construction. $\square$

4 Proof nets of MMLL

In this section we remind the main results about proof nets in MMLL, mainly the relations between proof nets and proofs in the sequent calculus with order varieties [1]. Then we introduce a new definition of proof nets with dependencies and analyze its connections with the sequent calculus. First we remind a definition of the proof nets in MMLL that is based on the notion of trip in a proof structure.

Definition 10 *The* links *of MMLL are the following graphs, the vertices of which are labelled with MMLL formulae:*



Axiom-links have two conclusions and no premisse. Cut links have two premisses and no conclusion. In the other cases, A and B are respectively the first and second premisse and the third formula is the conclusion.

Definition 11 *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 premisse of at most one link. An occurrence that is not a premisse 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) formulae.

Definition 12 *For any link l of a proof structure π, one defines a set $S(l)$ of partial functions, called* switching positions*, given in figure 4.*
Any function s such that, for any l of π, $s(l) \in S(l)$, is called a switching of π.
A switching for π is $\lhd_3$-free if for every $\lhd$-link, $s(l) \neq \lhd_3$.

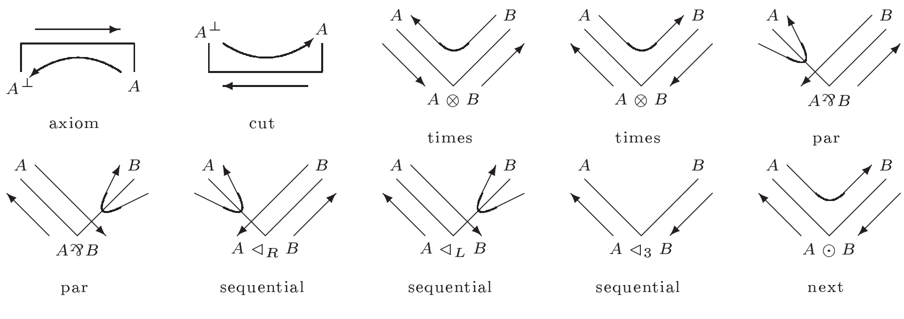


Fig. 4. Switching positions for MMLL

Definition 13 *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* *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 14 *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 formula A appears twice: one time decorated with $\downarrow$, another time with $\uparrow$. A cycle v in $s(\pi)$ is said* bilateral *if v has not the form $A^x, .., B^y, .., A^{\overline{x}}, .., B^{\overline{y}}, .., A^x$ where A and B are formulae occurrences of π.*

Theorem 15 *Let π be a proof structure of MMLL, then π is a proof net if and only if for any switching s of π : a) there exists exactly one cycle σ in $s(\pi)$, b) σ contains all the conclusions, c) σ is bilateral.*

Theorem 16 *([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 $\parr$) 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.*

This last result gives a characterization of a MMLL proof net π from its commutative translation π^* in MLL. The interest is that we can benefit of results about the construction of proof nets in MLL including some refinements or specializations.

4.1 Proof nets and order varieties

Definition 17 *If π is a proof net of MMLL and s is a switching for π, then the oriented graph with vertices the conclusions of π and an oriented edge from a conclusion A to a conclusion B iff there is no conclusion between $B^{\uparrow}$ and $A^{\downarrow}$ in the unique cycle σ in $s(\pi)$, is an oriented cycle called the cycle of the conclusions in $s(\pi)$*

Definition 18 *Let π be a proof net of MMLL with conclusion Γ,*
a) If s is a switching for π, $\alpha_{\pi,s}$ is the total order variety on Γ corresponding to the cycle of the conclusions in $s(\pi)$.
b) $\alpha_\pi = \natural(\bigcap_s \alpha_{\pi,s})$.

Definition 19 *To a proof D of conclusion $\vdash \Gamma \ \langle\alpha\rangle$ in sequent calculus, is associated in the obvious way a proof structure $\overline{D}$ with conclusion Γ.*

Theorem 20 (Adequacy) *If D is a sequential proof with conclusion $\vdash \Gamma \ \langle\alpha\rangle$, then $\overline{D}$ is a proof net and $\alpha = \alpha_{\overline{D}}$.*

Theorem 21 (Sequentialization) *Let π be a proof net with conclusion Γ. There exists a sequent calculus proof D, with conclusion $\vdash \Gamma\ \langle\alpha\rangle$ such that $\pi = \overline{D}$.*

Theorem 22 *$A \rightarrow_{\alpha_\pi} B$ iff for every switching s there is no conclusion between $B^\uparrow$ and $A^\downarrow$ in the unique cycle σ in $s(\pi)$.*

PROOF. Let π be a MMLL proof net with conclusions Γ, there exists a sequential proof with conclusion $\vdash \Gamma\ \langle\alpha_\pi\rangle$, where α_π is : $\alpha_\pi = \natural\left(\bigcap_s \alpha_{\pi,s}\right)$, where $\alpha_{\pi,s}$ is the order variety corresponding to the cycle of conclusions in the unique cycle σ in $s(\pi)$. $\square$

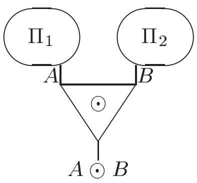
4.2 A new definition of MMLL proof nets

In this definition, we attach to each proof net Π a set of pairs of conclusions, called a set of dependencies $Dep(\Pi)$. This set is inductively built in parallel with the proof net structures (cf. construction rules).
In the rest of the paper $|Dep(\Pi)|_A$ (resp. $|Dep(\Pi)|_{A,B}$) represents the set of dependencies built from π with cancellation of the dependencies including A (resp. A or B). Moreover $Conc(\Pi)$, with Π being a proof net, is the set of conclusions of Π.

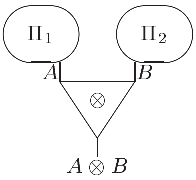
Definition 23 *The MMLL proof nets with dependencies are inductively defined in the following way:*

- *if Π is an axiom-link*  *then the set $Dep(\Pi)$ is built as follows:*

$$Dep(\Pi) = \left\{ \begin{array}{c} A \rightarrow A^\perp \\ A^\perp \rightarrow A \end{array} \right\}$$

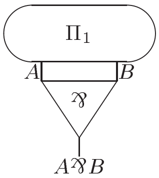
- *if Π is*  *the set $Dep(\Pi)$ is built from $Dep(\Pi_1)$ and $Dep(\Pi_2)$ as follows:*

$$\begin{aligned} Dep(\Pi) = \ & |Dep(\Pi_1)|_A \cup |Dep(\Pi_2)|_B \cup \\ & \{E \rightarrow F | E \in Conc(\Pi_1) \setminus A, F \in Conc(\Pi_2) \setminus B, E \rightarrow A \in Dep(\Pi_1), \\ & B \rightarrow F \in Dep(\Pi_2)\} \cup \\ & \{A \odot B \rightarrow E | E \in Conc(\Pi_1) \setminus A, A \rightarrow E \in Dep(\Pi_1)\} \cup \\ & \{F \rightarrow A \odot B | F \in Conc(\Pi_2) \setminus B, F \rightarrow B \in Dep(\Pi_2)\} \cup \\ & \{E \rightarrow A \odot B | E \in Conc(\Pi_1) \setminus A, E \rightarrow A \in Dep(\Pi_1), |Conc(\Pi_2)| = 1\} \cup \\ & \{A \odot B \rightarrow F | F \in Conc(\Pi_2) \setminus B, B \rightarrow F \in Dep(\Pi_2), |Conc(\Pi_1)| = 1\} \end{aligned}$$

- *if* Π *is* 

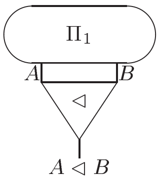
the set $Dep(\Pi)$ *is built from* $Dep(\Pi_1)$ *and* $Dep(\Pi_2)$ *as follows:*

$$\begin{aligned}
Dep(\Pi) \ = \ & |Dep(\Pi_1)|_A \cup |Dep(\Pi_2)|_B \cup \\
& \{E \to F | E \in Conc(\Pi_1) \setminus A, F \in Conc(\Pi_2) \setminus B, E \to A \in Dep(\Pi_1), \\
& B \to F \in Dep(\Pi_2)\} \cup \\
& \{A \otimes B \to E | E \in Conc(\Pi_1) \setminus A, A \to E \in Dep(\Pi_1)\} \cup \\
& \{F \to A \otimes B | F \in Conc(\Pi_2) \setminus B, F \to B \in Dep(\Pi_2)\} \cup \\
& \{F \to E | E \in Conc(\Pi_1) \setminus A, F \in Conc(\Pi_2) \setminus B, A \to E \in Dep(\Pi_1), \\
& F \to B \in Dep(\Pi_2)\} \cup \\
& \{A \otimes B \to F | F \in Conc(\Pi_2) \setminus B, B \to F \in Dep(\Pi_2)\} \cup \\
& \{E \to A \otimes B | E \in Conc(\Pi_1) \setminus A, E \to A \in Dep(\Pi_1)\}
\end{aligned}$$

- *if* Π *is* 

the set $Dep(\Pi)$ *is built from* $Dep(\Pi_1)$ *as follows:*

$$\begin{aligned}
Dep(\Pi) \ = \ & Dep(\Pi_1)[A \parr B/A, A \parr B/B] \\
& \{E \to F | E, F \in Conc(\Pi_1) \setminus \{A, B\}, \ E \to A, A \to F \in Dep(\Pi_1)\} \cup \\
& \{E \to F | E, F \in Conc(\Pi_1) \setminus \{A, B\}, \ E \to B, B \to F \in Dep(\Pi_1)\}
\end{aligned}$$

- *if* Π *is:* 

the set $Dep(\Pi)$ *is built from* $Dep(\Pi_1)$ *as follows:*

if $B \to A$ *is in all the maximal cycles in* $Dep(\Pi_1)$*, then we have*

$$Dep(\Pi) \ = \ Dep(\Pi_1)[A \lhd B/A, A \lhd B/B]$$

In fact, the construction of the dependencies in this inductive definition is similar to the one for the sequent calculus given in figure 3. The only syntactical differences come from the differences between sequents and proof nets in terms of notations. As in the sequent calculus, when one forgets the dependencies sets, the given inductive definition is exactly the one for the proof nets of MLL seeing $\odot$ and $\lhd$ as, respectively, $\otimes$ and $\parr$.

Let us comment this definition. The construction of $Dep(\Pi)$ is more complex for $\otimes$ than for $\odot$ because we extend the dependencies like for $A \odot B$ but also for $B \odot A$. In the case of $\parr$, we are obliged to apply the transitivity of the dependencies because the substitution of A and B by $A \parr B$ make lost the relationships between A and B. In the case of $\lhd$, the dependencies are the same than for $\parr$, except that, because of the condition, we know that $B \to A \in Dep(\Pi_1)$ and thus the application of transitivity is not necessary.

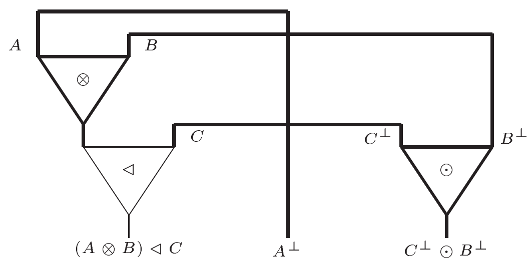
Theorem 24 *A proof structure π can be constructed using this inductive definition iff π is a proof net of MMLL.*

PROOF. From theorem 16, a proof structure π is a proof net iff : i) π^* is a proof net of MLL, ii) 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.
First, if we ignore the dependencies set, the given inductive definition is exactly the one for the proof nets of MLL seeing $\odot$ and $\lhd$ as, respectively, $\otimes$ and $⅋$. Thus, every structure π built using the inductive definition is such that π^* is a proof net of MLL. Conversely, if π is a proof structure such that π^* is a proof net of MLL, then π can be constructed iff the condition on the application of the $\lhd$-rule is verified for each $\lhd$-link in π.

Let π be a proof structure built using our inductive definition, and $\mathcal{D}$ be the set of dependencies associated to π. In theorem 26, we prove that the maximal cycles of $\mathcal{D}$ correspond exactly to the cycles of conclusions in π. Thus, the condition on the application of the $\lhd$-rule in the inductive definition insures for every $\lhd$-link $A \lhd B$ and every switching s in π, there is no conclusion between $B^\uparrow$ and $A^\downarrow$ in the unique cycle in $s(\pi)$. By induction, we can prove the inner parts of $\lhd$-link do not overlap. For a detailed proof of this point we can refer to [8].
Finally, we prove that a proof structure built using our inductive definition is a proof net of MMLL. Moreover, a proof net of MMLL can be built using our inductive definition. This is a direct consequence of theorem 16, since the maximal cycles exactly correspond to the cycles of conclusions. Indeed, the condition on the inner parts of $\lhd$-links ensures that the condition on the application of the $\lhd$-rule is verified. $\square$

4.2.1 An example

Here is a proof structure for the sequent $\vdash (A \otimes B) \lhd C, A^\perp, C^\perp \odot B^\perp$.



$$Dep(A \otimes B, C, A^\perp, C^\perp \odot B^\perp) = \left\{ \begin{array}{rlrl} C & \to A \otimes B & C & \to A^\perp \\ A \otimes B & \to A^\perp & A^\perp & \to A \otimes B \\ A^\perp & \to C^\perp \odot B^\perp & A \otimes B & \to C^\perp \odot B^\perp \\ C^\perp \odot B^\perp & \to C & & \end{array} \right\}$$

There are two maximal cycles in $Dep(A \otimes B, C, A^\perp, C^\perp \odot B^\perp)$, namely $C^\perp \odot B^\perp \to C \to A \otimes B \to A^\perp \to C^\perp \odot B^\perp$ and $C^\perp \odot B^\perp \to C \to A^\perp \to A \otimes B \to C^\perp \odot B^\perp$. In fact, $C \to A \otimes B$ belongs to the first cycle, but not to the second cycle. Thus, the $\lhd$ rule is not applicable. As the choice for the axiom links to build is unique, we cannot perform any backtracking, and then cannot build a proof net for the given sequent. Thus, we can deduce that the given sequent is not provable.

4.3 Properties of proof nets

Definition 25 *Let $\mathcal{C}$ be an oriented cycle, we define $\overline{\mathcal{C}}$ as the inverse of $\mathcal{C}$, i.e., for every couple of vertices (A, B), $A \to B \in \mathcal{C}$ iff $B \to A \in \overline{\mathcal{C}}$.*

Theorem 26 *Let π be a proof net. $\mathcal{C}$ is a maximal cycle in $Dep(\pi)$ iff $\overline{\mathcal{C}}$ is a cycle of conclusions in π.*

PROOF. The main point is to notice that the construction of a dependencies set is made in respect with the switching positions of the various links. Thus, when focusing on maximal cycles, we actually build the cycles of conclusions of the proof net. We proceed by induction on the structure of proof nets. □

From a proof net of MMLL, we can easily build a proof in the sequent calculus with dependencies, and conversely, we can also build a proof net from a proof in the sequent calculus with dependencies.

Theorem 27 (Adequacy) *If D is a sequential proof with conclusion $\vdash \Gamma\ \mathcal{D}$, then $\overline{D}$ is a proof net and $Dep\left(\overline{D}\right) = \mathcal{D}$.*

PROOF. Proof similar to the one of MLL, by tacking dependency relations into account. □

Theorem 28 (Sequentialization) *Let π be a proof net with conclusion Γ. There exists a sequent calculus proof D, with conclusion $\vdash \Gamma\ Dep(\pi)$ such that $\pi = \overline{D}$.*

PROOF. Proof similar to the one of MLL, by tacking dependency relations into account. □

5 Proof search and dependencies in MMLL

We have seen that both sequent calculus (with order varieties and with dependencies) relies on MLL. However, we have a procedure for automatic proof nets construction in MLL [7] that we can adapt to design a procedure for proof net construction in MMLL.

5.1 Principles of construction

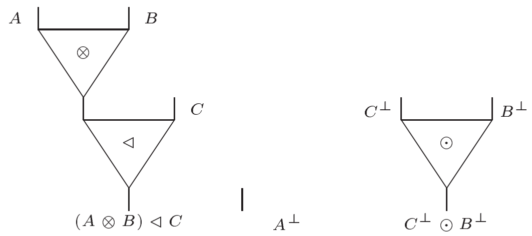
This procedure is based on principles that are similar to the ones of our algorithm for MLL proof nets construction.
From the inductive definition of proof nets, we notice that from two disjoint proof nets, we can build a new proof net by fusion and extension with a $\otimes$-link or a $\odot$-link and that from a given proof net we can build a new proof net by extension with a $⅋$-link or a $\triangleleft$-link. At this step, commutative and non-commutative connectives have the same behaviour during this construction but their semantical differences are reflected in the set of dependencies attached to proof structures.

In fact, it means that we apply the algorithm for MLL on the commutative translation of the formula to prove and in addition we build set of dependencies that are used to test the condition about the validity of the application of the $\triangleleft$ rule. Therefore, our algorithm considers the decomposition tree of the given sequent, that will be used as a frame to guide the proof net search. In addition, 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 $\otimes$ or $\odot$-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 $⅋$ or $\triangleleft$-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 $⅋$ or $\triangleleft$-link (in the decomposition tree) are conclusions of two different nets in $\mathcal{R}$ then we postpone the extension of such a link. With respect to the permutability results in linear sequent calculus [9], we have to always apply, if possible, a $\otimes$ or $\odot$ (extension) rule before a $⅋$ or $\triangleleft$ (extension) rule.

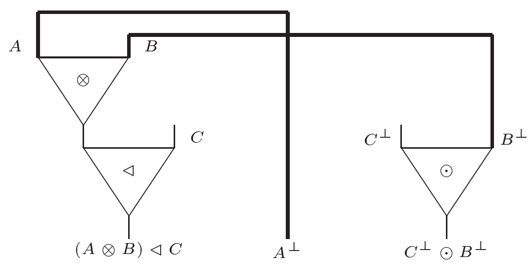
The algorithm is proved complete and correct using the arguments of the MLL case and the new proof net definition. A detailed description of the algorithm and its proofs of correction and completeness will be given in the full version.

5.2 An example

We illustrate the algorithm with the sequent $\vdash (A \otimes B) \lhd C, A^{\perp}, C^{\perp} \odot B^{\perp}$. We first start by building the decomposition for the sequent to prove.

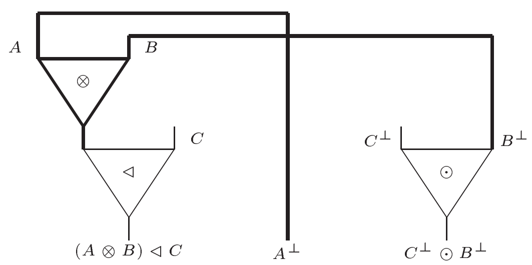


We then build the two elementary proof nets corresponding to the axioms $\vdash A^{\perp}, A$ and $\vdash B^{\perp}, B$.



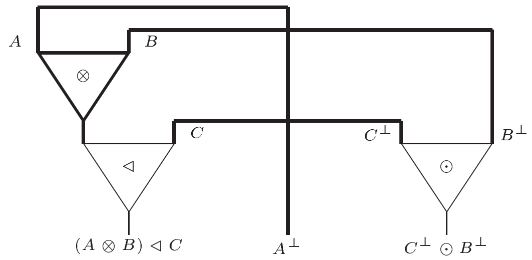
$$Dep(A - A^{\perp}) = \left\{ \begin{array}{c} A \to A^{\perp} \\ A^{\perp} \to A \end{array} \right\} \; and \;\; Dep(B - B^{\perp}) = \left\{ \begin{array}{c} B \to B^{\perp} \\ B^{\perp} \to B \end{array} \right\}$$

We can now apply the rule for the $\otimes$-link.



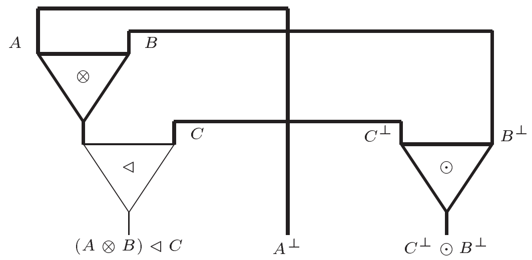
$$Dep(A^{\perp}, A \otimes B, B^{\perp}) = \left\{ \begin{array}{rclrcl} B^{\perp} & \to & A \otimes B & B^{\perp} & \to & A^{\perp} \\ A \otimes B & \to & A^{\perp} & A^{\perp} & \to & A \otimes B \\ A^{\perp} & \to & B^{\perp} & A \otimes B & \to & B^{\perp} \end{array} \right\}$$

We now choose the literal $C^{\perp}$, and try to branch it to the free literal C. We obtain a new elementary proof net corresponding to the axiom $\vdash C^{\perp}, C$.



$$Dep(C, C^\perp) = \left\{ \begin{array}{r} C^\perp \to C \\ C \to C^\perp \end{array} \right\}$$

We apply the rule for the $\odot$-link, and thus obtain a new partial proof net.



$$Dep(A \otimes B, C, A^\perp, C^\perp \odot B^\perp) = \left\{ \begin{array}{rr} C \to A \otimes B & C \to A^\perp \\ A \otimes B \to A^\perp & A^\perp \to A \otimes B \\ A^\perp \to C^\perp \odot B^\perp & A \otimes B \to C^\perp \odot B^\perp \\ C^\perp \odot B^\perp \to C & \end{array} \right\}$$

There are two maximal cycles in $Dep(A \otimes B, C, A^\perp, C^\perp \odot B^\perp)$ that are $C^\perp \odot B^\perp \to C \to A \otimes B \to A^\perp \to C^\perp \odot B^\perp$ and $C^\perp \odot B^\perp \to C \to A^\perp \to A \otimes B \to C^\perp \odot B^\perp$. In fact, $C \to A \otimes B$ belongs to the first one, but not to the second one. Thus, the $\lhd$ rule is not applicable. As the choice for the axiom links to build is unique, we cannot perform any backtracking, and then cannot build a proof net for the given sequent. Thus, we can deduce that the given sequent is not provable.

6 Dependencies with constraints

In order to design efficient procedures for proof (net) construction, we aim to optimize the construction of maximal cycles. In the sequent calculus with dependencies, we have to consider a special condition on the application of the $\lhd$-rule, namely $\frac{\vdash \Gamma, A, B \;\; \mathcal{D}_1}{\vdash \Gamma, A \lhd B \;\; \mathcal{D}} \lhd$, in order to take into account the non-commutativity of $\odot$ and $\lhd$. This condition is $B \to A$ *is in all the maximal cycles in* $\mathcal{D}_1$. Thus, one needs to extract the maximal cycles in the graph of dependencies $\mathcal{D}_1$, this step being a source of potential complexity in the proof search. For that, we extend our calculus on dependencies in order to directly check this condition about the $\lhd$-rule application, from the dependencies set.

A solution consists in adding guards on dependencies in order to avoid the construction of maximal cycles. These guards are in fact boolean formulae associated to dependencies. A valuation for the boolean variables will correspond to a maximal cycle : given a valuation v, we only keep the dependencies $A \xrightarrow{x} B$ such that the formulae x equals to 1 for the valuation v. Thus, to check if a dependency is in all the maximal cycle, we only need to check the validity of the boolean formula associated to it.
Finally, if we consider the $\lhd$-rule of the sequent calculus $\frac{\vdash \Gamma, A, B \ \mathcal{D}_1}{\vdash \Gamma, A \lhd B \ \mathcal{D}} \lhd$, the condition of its application can be reformulated by : *from* $\mathcal{D}_1$, *we can deduce* $B \xrightarrow{1} A$ (denoted $\mathcal{D}_1 \vdash B \xrightarrow{1} A$). The boolean value, associated to a dependency relation, is then computed during the construction of the dependencies set.

This new sequent calculus will be defined in the full version of this paper, and its proof of equivalence with the initial sequent calculus with dependencies will be given. We will also illustrate how constraints can be used and solved in order to determinate the validity of the new condition of the $\lhd$ rule. Moreover we will propose a version of the proof nets calculus with such constraints and present their main results (equivalence with the previous system, ability to deal with efficient proof-search).

7 Conclusion

In this paper, we propose new calculi for the multiplicative fragment of Mixed Linear Logic (MMLL), that are are based on the definition of dependency-relations. They provide a new proof-theoretical formulation of MMLL in the spirit of [5] and moreover the use of dependencies (or constraints) is a natural way to translate some proof-theoretical results of MMLL into simple and tractable automated proof-search procedures, that are based on the construction of semantical structures like proof nets [6,11]. Then, we provide an algorithm for proof net construction in MMLL that builds in parallel a proof structure and a set of dependencies on which we have to verify some validity conditions. Moreover, we propose refinements of both calculi that include dependencies with constraints in order to simplify their implementations. In further works, we will compare our algorithm with the previous ones we have recently developed [8]. In addition, we will show how to adapt our approach, that is based on proof nets, for proof (net) verification. Our new MMLL proof net definition provides a new correctness criterion for MMLL and we will discuss its complexity and compare it with the recent quadratic correctness criterion proposed for MMLL in [13].

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] G. Bellin and P.J. Scott. On the π-calculus and linear logic. *Theoretical Computer Science*, 135(1):11–66, 1994.

[5] T. Bräuner and V. De Paiva. A formulation of linear logic based on dependency-relations. In *Computer Science Logic, CSL'97, LNCS 1414*, pages 129–148. Springer Verlag, 1997.

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

[7] 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.

[8] D. Galmiche and J.M. Notin. Proof-search and proof nets in mixed linear logic. *Electronic Notes in Theoretical Computer Science*, 37, 2000.

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

[10] 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.

[11] 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.

[12] N. Kobayashi and A. Yonezawa. Asynchronous communication model based on linear logic. *Formal Aspects of Computing*, 3:279–294, 1994.

[13] V. Mogbil. Quadratic correctness criterion for non commutative logic. In *Computer Science Logic, CSL'01, LNCS*, Paris, France, September 2001.

[14] P. Ruet. Non-commutative logic II : sequent calculus and phase semantics. *Math. Struct. in Comp. Science*, 10:277–312, 2000.

[15] 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.