Cyclic Multiplicative-Additive Proof Nets of Linear Logic with an Application to Language Parsing

Vito Michele Abrusci and Roberto Maieli(✉)

Department of Mathematics and Physics, Roma Tre University,
Largo San Leonardo Murialdo, 1, 00146 Rome, Italy
{abrusci,maieli}@uniroma3.it

Abstract. This paper concerns a logical approach to natural language parsing based on proof nets (PNs), i.e. de-sequentialized proofs, of linear logic (LL). It first provides a syntax for proof structures (PSs) of the cyclic multiplicative and additive fragment of linear logic (CyMALL). A PS is an oriented graph, weighted by boolean monomial weights, whose conclusions Γ are endowed with a cyclic order σ. Roughly, a PS π with conclusions $\sigma(\Gamma)$ is correct (so, it is a proof net), if any slice $\varphi(\pi)$, obtained by a boolean valuation φ of π, is a multiplicative (CyMLL) PN with conclusions $\sigma(\Gamma_r)$, where Γ_r is an additive resolution of Γ, i.e. a choice of an additive subformula for each formula of Γ. The correctness criterion for CyMLL PNs can be considered as the non-commutative counterpart of the famous Danos-Regnier (DR) criterion for PNs of the pure multiplicative fragment (MLL) of LL. The main intuition relies on the fact that any DR-switching (i.e. any correction or test graph for a given PN) can be naturally viewed as a seaweed, that is, a rootless planar tree inducing a cyclic order on the conclusions of the given PN. Unlike the most part of current syntaxes for non-commutative PNs, our syntax allows a sequentialization for the full class of CyMALL PNs, without requiring these latter to be cut-free.

One of the main contributions of this paper is to provide a characterization of CyMALL PNs for the additive Lambek Calculus and thus a geometrical (non inductive) way to parse sentences containing words with syntactical ambiguity (i.e., with polymorphic type).

1 Introduction

Proof nets (PNs) are one of the most innovative inventions of linear logic (LL, [7]): they are used to represent demonstrations in a geometric (i.e., non inductive) way, abstracting away from the technical bureaucracy of sequential proofs. Proof nets quotient classes of derivations that are equivalent up to some irrelevant permutations of inference rules instances.

In this spirit, we present a syntax for PNs of the cyclic multiplicative and additive fragment of linear logic (CyMALL, Sect. 1.1). This syntax, like the original one of Girard [8], is based on weighted (by boolean monomials) proof structures with explicit binary contraction links (Sect. 2). The conclusions Γ (i.e.,

A. Foret et al. (Eds.): FG 2015/2016, LNCS 9804, pp. 43–59, 2016.
DOI: 10.1007/978-3-662-53042-9_3

a sequence of formula occurrences) of any PS are endowed with a cyclic order σ on Γ. Naively, a CyMALL PS π with conclusions $\sigma(\Gamma)$ is correct if, for any slice $\varphi(\pi)$, obtained by a boolean valuation φ of π, there exists an additive resolution (i.e., a multiplicative restriction of $\varphi(\pi)$) that is a CyMLL PN with conclusion $\sigma(\Gamma_r)$, where Γ_r is an additive resolution of Γ (i.e. a choice of an additive sub-formula for each formula of Γ). In turn, the correctness criterion for CyMLL PNs can be considered as the non-commutative counterpart of the famous Danos-Regnier (DR) criterion for proof nets of linear logic (see [5,6]). The main intuition relies on the fact that any *DR-switching for a PS* (i.e. any correction or test graph, obtained by mutilating one premise of each disjunction ∇-link) can be naturally viewed as a rootless planar tree, called a *seaweed*, inducing a *cyclic ternary relation* on the conclusions of the given proof structure.

Unlike some previous syntaxes for non-commutative logic, like e.g., [3,13], this new syntax admits a *sequentialization* (i.e., a correspondence with sequential proofs) for the full class of CyMLL PNs including those ones with cuts. Actually, the presence of cut links is "rather tricky" in the non-commutative case, since cut links are not equivalent, from a topological point of view, to tensor links (like in the commutative MLL case): indeed, tensor links make new conclusions appear that may disrupt the original (i.e., in presence of cut links) order of conclusions.

CyMALL PNs satisfy a simple (lazy) convergent cut-elimination procedure (Sect. 2.2) in Laurent-Maieli's style [12]: our strategy relies on the notion of *dependency graph of an eigen weight p* (Definition 9), that is, the smallest $\&_p$-box that must be duplicated in a commutative $\&_p/C$-cut reduction step [14]. Moreover, cut-reduction preserves PNs sequentialization (Sect. 2.3).

CyMALL can be considered as a conservative classical extension of Lambek Calculus (LC, see [1,11,17]) one of the ancestors of LL. The LC represents the first attempt of the so called *parsing as deduction*, i.e., parsing of natural language by means of a logical system. Following [4], in LC, parsing is interpreted as type checking in the form of theorem proving of Gentzen sequents. Types (i.e. propositional formulas) are associated to words in the lexicon; when a string $w_1...w_n$ is tested for grammaticality, the types $t_1, ..., t_n$ associated with the words are retrieved from the lexicon and then parsing reduces to proving the derivability of a one-sided sequent of the form $\vdash t_n^{\perp}, ..., t_1^{\perp}, s$, where s is the type associated with sentences. Moreover, forcing constraints on the Exchange rule, by e.g. allowing only *cyclic permutations* over sequents of formulas, gives the required computational control needed to view theorem proving as parsing in Lambek Categorial Grammar style. Anyway, LC parsing presents some syntactical ambiguity problems; actually, there may be:

1. (*non canonical proofs*) more than one cut-free proof for the same sequent;
2. (*lexical polymorphism*) more than one type associated with a single word.

Now, proof nets are commonly considered an elegant solution to the first problem of representing canonical proofs; in this sense, in Sect. 3, we give an *embedding* of extended MALL Lambek Calculus into Cyclic MALL PNs. Concerning the second problem, in Sect. 4, we propose a parsing approach based on CyMALL PNs that could be considered a step towards a proof-theoretical solution to

the problem of lexical polymorphism. Technically, CyMALL proof nets allow to manage formulas superposition (types polymorphism) by means of the additive &-links, or dually, $\oplus$-links. By means of Lambek CyMALL PNs, we propose the parsing of some sentences, suggested by [18], which make use of polymorphic words; naively, when a word has two possible formulas A and B assigned, then we can combine (or super-pose) these into a single additive formula $A\&B$.

1.1 The Cyclic MALL Fragment of Linear Logic

We briefly recall the necessary background of the *Cyclic MALL fragment* of LL, denoted *CyMALL*, without units (see [1]). We arbitrarily assume literals $a, a^\perp, b, b^\perp, ...$ with a polarity: *positive* (+) for atoms, $a, b, ...$ and *negative* (−) for their duals $a^\perp, b^\perp ....$ A *formula* is built from literals by means of the two groups of connectives: *negative*, ∇ ("par") and & ("with") and *positive*, $\otimes$ ("tensor") and $\oplus$ ("plus"). For these connectives we have the following De Morgan laws: $(A \otimes B)^\perp = B^\perp \nabla A^\perp$, $(A \nabla B)^\perp = B^\perp \otimes A^\perp$, $(A\&B)^\perp = B^\perp \oplus A^\perp$, $(A \oplus B)^\perp = B^\perp \& A^\perp$. A CyMALL (resp., CyMLL) proof is any derivation tree built by the following (resp., by only *identities and multiplicative*) inference rules where sequents Γ, Δ are sequences of formula occurrences endowed with a *total cyclic order* (or *cyclic permutation*).

identities:
$$\frac{}{\vdash A, A^\perp}\ axiom \qquad\qquad \frac{\vdash \Gamma, A \qquad A^\perp \Delta}{\vdash \Gamma, \Delta}\ cut$$

multiplicatives:
$$\frac{\vdash \Gamma, A \qquad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta}\ \otimes \qquad\qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \nabla B}\ \nabla$$

additives:
$$\frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A\&B}\ \& \qquad\qquad \frac{\vdash \Gamma, A_i}{\vdash \Gamma, A_1 \oplus_i A_2}\ \oplus_{i=1,2}$$

A *total cyclic order* can be thought as follows; consider a set of points of an oriented circle; the orientation induces a total order on these points as follows: if a, b and c are three distinct points, then b is either between a and c ($a < b < c$) or between c and a ($c < b < a$). Moreover, $a < b < c$ is equivalent to $b < c < a$ or $c < a < b$.

Definition 1 (total cyclic order). *A* total cyclic order *is a pair* (X, σ) *where* X *is a set and* σ *is a ternary relation over* X *satisfying the following properties:*

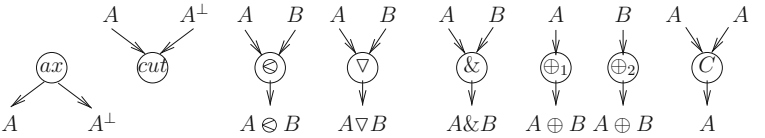
1. $\forall a, b, c \in X, \sigma(a, b, c) \rightarrow \sigma(b, c, a)$ (cyclic),
2. $\forall a, b \in X, \neg\sigma(a, a, b)$ (anti-reflexive),
3. $\forall a, b, c, d \in X, \sigma(a, b, c) \wedge \sigma(c, d, a) \rightarrow \sigma(b, c, d)$ (transitive),
4. $\forall a, b, c \in X, \sigma(a, b, c) \vee \sigma(c, b, a)$ (total).

Negative (or *asynchronous*) connectives correspond to *true determinism* in the way we apply bottom-up their corresponding inference rules. In particular, observe that Γ must appear as the same context (with the same order) in both premises of the &-rule. Positive (or *synchronous*) connectives correspond to *true*

non-determinism in the way we apply, bottom-up, their corresponding rules; there is no deterministic way to split, bottom up, the context (Γ, Δ) in the $\otimes$-rule; similarly, there not exist a deterministic way to select, bottom up, $\oplus_1$ or $\oplus_2$-rule.

2 Cyclic MALL Proof Structures

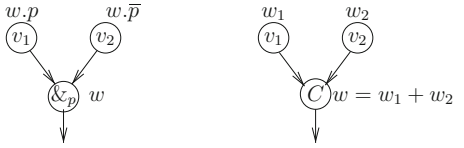
Definition 2 (CyMALL proof structure). *A* CyMALL proof structure (PS) *is an oriented graph π whose edges (resp., nodes) are labeled by formulas (resp., by connectives) of CyMALL and built by juxtaposing the following special graphs, called* links*, in which incident (resp., emergent) edges are called* premises *(resp.,* conclusions*):*



In a PS each premise (resp., conclusion) of a link must be conclusion (resp., premise) of exactly (resp., at most) one link. We call conclusion of a PS *any emergent edge that is not premises of any link. We call* CyMLL proof structure*, any PS built by only means of axioms, cut and multiplicative links ($\otimes, \nabla$).*

Definition 3 (Girard CyMALL proof structure). *A* Girard proof structure (GPS) *is a PS with* weights *associated as follows (a* weights assignment*):*

1. *first we associate a* boolean variable*, called* eigen weight *p, to each &-node (eigen weights are supposed to be different);*
2. *then we associate a* weight*, a product of (negation of) boolean variables $(p, \overline{p}, q, \overline{q}...)$ to each node, with the constraint that two nodes have the same weight if they have a common edge, except when the edge is the premise of a & or C-node; in these cases we do like below:*



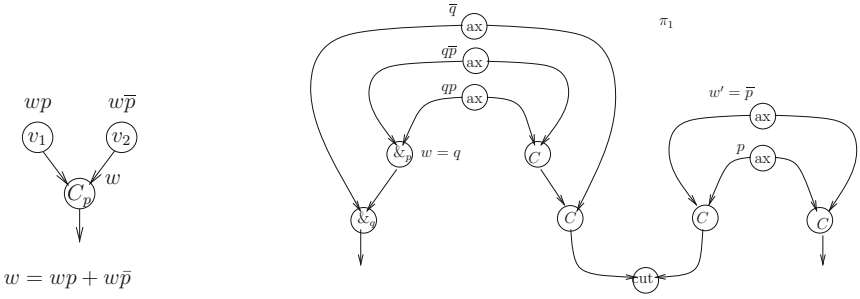
if p does not occur in w with $w_1.w_2 = 0$

3. *a* conclusion node *has weight* 1*;*
4. *if w is the weight of a &-node, with eigen weight p, and w' is a weight depending on p and appearing in the proof structure then $w' \leq w$.*

A weight w *depends on* the eigen weight p if p or $\bar{p}$ occurs in w. A node L with weight w *depends on* the eigen weight p if w depends on p or L is a C-node and one of the weights just above it depends on p.

Remark 1. Observe that:

1. since weights associated to a PS are products (*monomials*) of the Boolean algebra generated by the eigen weights associated to a proof structure, then, for each weight w associated to a binary contraction node, there exists a unique eigen weight p that *splits* w into $w_1 = wp$ *and* $w_2 = w\overline{p}$. We sometimes index a C-link with its *toggling variable* p, see the next left hand side picture;
2. the graph π_1 (the next r.h.s. picture) is not a GPS since it violates condition 4 of Definition 3; indeed, if $w = q$ is the weight of the $\&_p$-link and $w' = \overline{p}$ is a weight depending on p and appearing in the proof-structure then $\overline{p} \not\leq q$.



2.1 Correctness

Definition 4 (slices, switchings, resolutions). *Let π be a CyMALL GPS.*

- *A* valuation *φ of π is a function from the set of all weights of π into $\{0, 1\}$.*
- *Fixed a valuation φ for π, the* slice *$\varphi(\pi)$ is the graph obtained from π by keeping only those nodes with weight* 1 *together with their incident edges.*
- *Fixed a slice $\varphi(\pi)$ a* multiplicative switching *S for π is the non-oriented graph $S^m_\varphi(\pi)$ built on the nodes and edges of $\varphi(\pi)$ with the modification that for each ∇-node we take only one premise (*left/right switch*).*
- *Fixed a slice $\varphi(\pi)$ an* additive switching*, denoted $S^a_\varphi(\pi)$ is a multiplicative switching $S^m_\varphi(\pi)$ for π, in which for each $\&_p$-node we erase the (unique) premise in $\varphi(\pi)$ and we add an oriented edge, called* jump*, from the $\&_p$-node to an link L whose weight depends on the eigen weight p.*
- *An* additive resolution *$\varphi_r(\pi)$ for a slice $\varphi(\pi)$ is the graph obtained by replacing in $\varphi(\pi)$ each unary link L (a link that, possibly, after the valuation has a single premise) by a single edge that is the (unique) premise of L. In particular, each conclusion of $\varphi_r(\pi)$ will be labeled by a multiplicative (CyMLL) formula.*

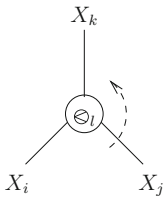
We call *additive resolution of a CyMALL sequent Γ* what remains of Γ after deleting one of the two sub-formulas in each additive (sub)formula of Γ.

In the following we characterize, by a *correctness criterion*, those CyMALL GPSs corresponding to proofs. This correctness criterion (Definition 7) is defined in terms of the correctness of CyMLL PSs (Definition 6). There exist several syntaxes for CyMLL proof nets; here we adopt the syntax of [2] inspired to [13].

Definition 5 (seaweeds). *Assume π is a CyMLL PS with conclusions Γ and assume $S(\pi)$ is an acyclic and connected multiplicative switching for π; $S(\pi)$ is*

the rootless planar tree whose nodes are labeled by $\Subset$-nodes, and whose leaves $X_1, ..., X_n$ (with $\Gamma \subseteq X_1, ..., X_n$) are the terminal (pending) edges of $S(\pi)$; $S(\pi)$ is a ternary relation, called a seaweed, *with* support $X_1, ..., X_n$*; an ordered triple (X_i, X_j, X_k) belongs to the seaweed $S(\pi)$ iff:*

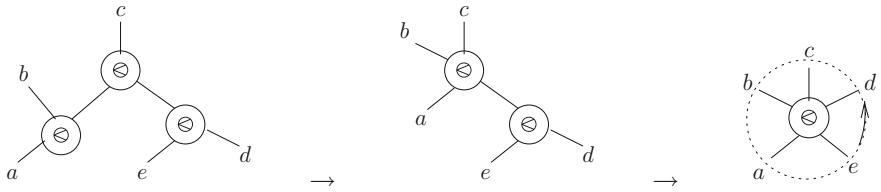
- *the intersection of the three paths X_iX_j, X_jX_k and X_kX_i is the node $\Subset_l$;*
- *the three paths $X_i\Subset_l, X_j\Subset_l$ and $X_k\Subset_l$ are in this cyclic order while moving anti-clockwise around the $\Subset_l$-node as below.*



If A is an edge of the seaweed $S(\pi)$, then $S_i(\pi) \downarrow^A$ is the restriction of the seaweed $S(\pi)$*, that is, the sub-graph of $S(\pi)$ obtained as follows:*

1. *disconnect the graph below (w.r.t. the orientation of π) the edge A.*
2. *delete the graph not containing A.*

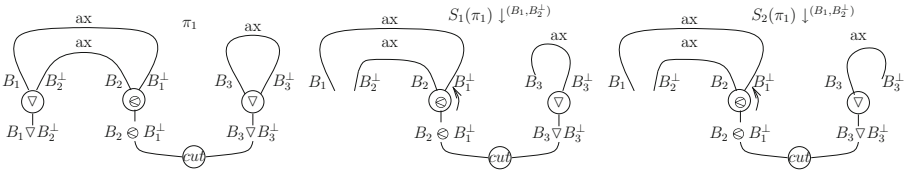
Fact 1 (seaweeds as cyclic orders). *Any seaweed $S(\pi)$ can be viewed as a cyclic total order (Definition 1) on its support $X_1, ..., X_n$; in other words, if a triple $(X_i, X_j, X_k) \in S(\pi)$, then $X_i < X_j < X_k$ are in cyclic order. Intuitively, we may contract a seaweed (by associating the $\Subset$-nodes) until it collapses into single n-ary $\Subset$-node with n pending edges (its support), like in the example below.*



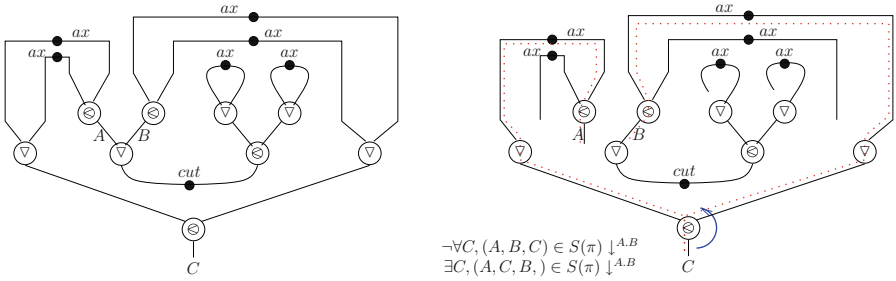
Definition 6 (CyMLL proof net). *A CyMLL PS π is* correct*, i.e. it is a CyMLL proof net (PN), iff:*

1. *π is a standard MLL PN, that is, any switching $S(\pi)$ is a connected and acyclic graph (therefore, $S(\pi)$ is a seaweed);*
2. *for any ∇-link $\frac{A \quad B}{A \nabla B}$ the triple (A, B, C) must occur in this cyclic order in any seaweed $S(\pi)$ restricted to A, B, i.e., $(A, B, C) \in S(\pi) \downarrow^{(A,B)}$, for all pending leaves C (if any) in the support of the restricted seaweed.*

Example 1 (CyMLL PSs). We give below an instance of CyMLL PN π_1 with its two restricted seaweeds, $S_1(\pi_1) \downarrow^{(B_1, B_2^\perp)}$ and $S_2(\pi_1) \downarrow^{(B_1, B_2^\perp)}$, both satisfying condition 2 of Definition 6.



Mellies's Counter-Example. Observe that, unlike what happens in the commutative MLL case, the presence of cut links is "quite tricky" in the non-commutative case, since cut links are not equivalent, from a topological point of view, to tensor links: these latter make appear new conclusions that may disrupt the original (i.e., in presence of cut links) order of conclusions. In particular, the *Mellies's proof structure*[1] below (see page 224 of [17]) is not correct according to our correctness criterion since there exists a $\frac{A \quad B}{A \nabla B}$ link and a switching $S(\pi)$ s.t. $\neg\forall C, (A, B, C) \in S(\pi) \downarrow^{(A,B)}$, contradicting condition 2 of Definition 6: following the crossing dotted lines in the next r.h.s. figure, you can easily verify $\exists C$ pending s.t. $(A, C, B) \in S(\pi) \downarrow^{(A,B)}$.

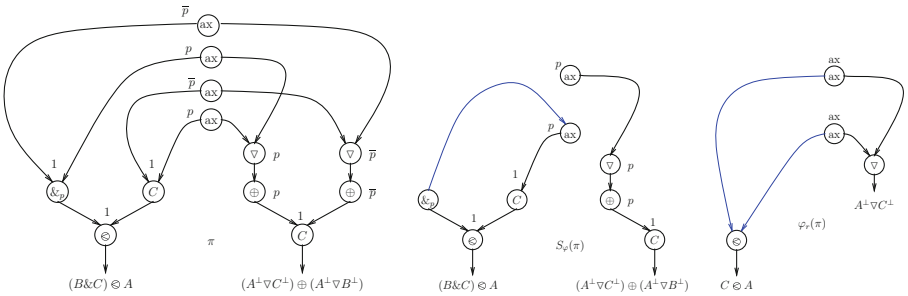


Definition 7 (CyMALL proof net). *We call* correct *(or* proof net, GPN*) any CyMALL GPS π s.t., its conclusions Γ are endowed with a cyclic order $\sigma(\Gamma)$ and for any valuation φ of π:*

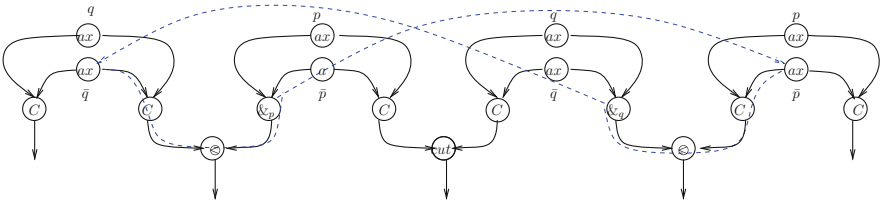
1. *each additive switching $S_\varphi(\pi)$ is an acyclic and connected graph (ACC);*
2. *there exists an additive resolution $\varphi_r(\pi)$ for $\varphi(\pi)$ that is a CyMLL PN with cyclic order conclusions $\sigma(\Gamma_r)$, where Γ_r is an additive resolution of Γ.*

Example 2 (CyMALL GPSs). Observe that the following proof structure π, on the left hand side, is not correct: actually, fixed a valuation φ s.t. $\varphi(p) = 1$, there exists an additive switching $S_\varphi(\pi)$ (with a jump) that is not ACC (see the center side figure). Nevertheless, any slice $\varphi(\pi)$ is ACC; for each slice $\varphi(\pi)$ there exists indeed an additive resolution $\varphi_r(\pi)$ that is a CyMLL PN like that one, on the rightmost hand side, with conclusions $C \otimes A, A^\perp \nabla C^\perp$. Observe, $\Gamma_r = (C \otimes A, A^\perp \nabla C^\perp)$ is an additive resolution of the conclusion of π, $\Gamma = (B \& C) \otimes A, (A^\perp \nabla C^\perp) \oplus (A^\perp \nabla B^\perp)$.

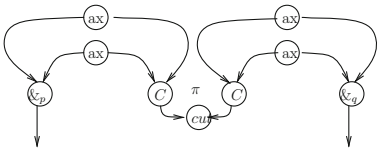
[1] This PS is considered as "a measure of the satisfiability degree" of correctness criteria of non-commutative logic: any "good" criterion should recognize this PS as uncorrect.



Similarly, the proof structure below is not correct: you can easily get an additive switching with a cycle like that one in (blue) dashed line.



Finally, the proof structure below is correct.

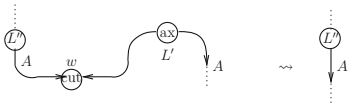


There currently exist other syntaxes for MALL PNs like the recent one by Hughes–van Glabbeek [10]. Unlike the Girard's one, this new syntax only works with "uniform proof structures", i.e., proof structures with only η-expanded axioms and with contraction links only immediately below the axiom links.

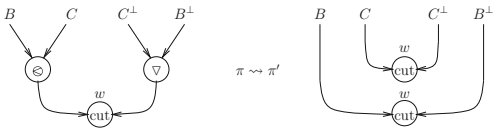
2.2 Cut Reduction

Definition 8 (ready cut reduction). *Let L be a cut link in a proof net π whose premises A and $A^{\perp}$ are conclusions of, resp., links L' and L'' with both of these different from contraction C. Then we define the result π' (called,* reductum*) of reducing a* ready cut *in π (called,* redex*), as follows:*

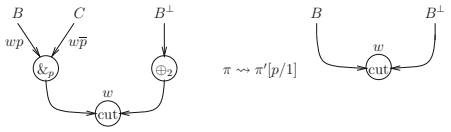
Ax*-cut: if L' (resp., L'') is an axiom link then π' is obtained by removing in π both formulas $A, A^{\perp}$ (as well as L) and giving to L'' (resp., to L') the other conclusion of L' (resp., L'') as new conclusion:*



($\parr/\otimes$)-cut: *if L' is a $\parr$-link with remises B and C and L'' is a $\otimes$-link with premises $C^\perp$ and $B^\perp$, then π' is obtained by removing in π both formulas A and $A^\perp$ as well as the cut link L together with L' and L'' and by adding two new cut links with, resp., premises B, $B^\perp$ and $C, C^\perp$, as follows:*



($\&/\oplus$)-cut: *if L' is a $\&_p$-link with premises B and C and L'' is a $\oplus_2$-link (resp., a $\oplus_1$-link) with premise $B^\perp$ (resp., $C^\perp$), then π' is obtained in three steps: first remove in π both formulas A, $A^\perp$ as well as the cut link L with L' and L'', then replace the eigen weight p by 1 (resp., p by 0) and keep only those links (vertexes and edges) that still have non-zero weight; finally we add a cut between B and $B^\perp$ (resp., between C and $C^\perp$) as below.*



Theorem 1 (stability of GPN under ready cut reduction). *Assume π is a GPN that reduces to π' in one step of ready cut reduction, then π' is a GPN.*

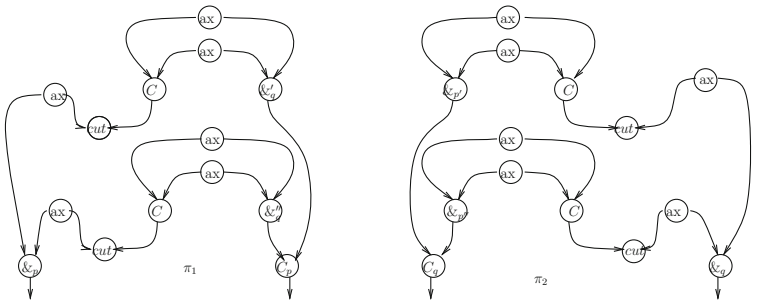
Proof. Stability of condition 1 of Definition 6 and condition 1 of Definition 7, under ready cut reduction, follows as a consequence of the next graph theoretical property (see pages 250–251 of [9]):

Property 1 (Euler-Poicaré invariance). Given a graph $\mathcal{G}$, then $(\sharp CC - \sharp Cy) = (\sharp V - \sharp E)$, where $\sharp CC$, $\sharp Cy$, $\sharp V$ and $\sharp E$ denotes the number of, respectively, connected components, cycles, vertexes and edges of $\mathcal{G}$.

Meanwhile, stability of condition 2 of Definition 6 (resp., condition 2 of Definition 7) follows simply by calculation.

The confluence problem - Reducing a cut involving a contraction link as (at least) one of its premises may lead to different reductum, depending on which sub-graph of the redex we decide to duplicate. For instance, as illustrated below, reducing the commutative cut of the last proof net of Example 2 leads either to π_1 or to π_2 (in the next picture), depending on which additive box, $\&_q$ or $\&_p$, we decide to duplicate. There is no a-priori way to make π_1 and π_2 "equal". Girard, in [8], did not give a solution for this problem which has been later provided by Laurent and Maieli in [12]. Here we present an original lazy commutative cut reduction that simplifies the latter: technically, our reduction relies on the notion of *dependency graph* (Definition 9), i.e. the smallest &-box needed

for duplication (see [14]). This cut reduction procedure preserves the notion of GPN (Theorems 1 and 2) and it is strong normalizing (Theorems 3 and 4).

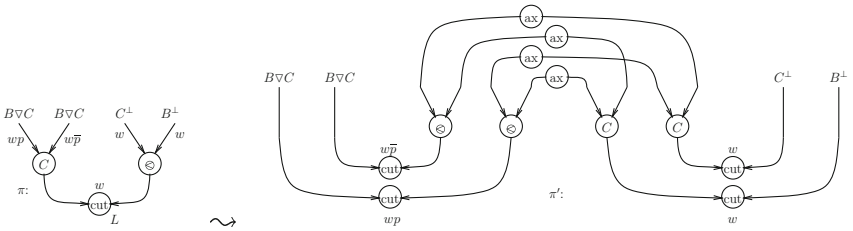


Definition 9 (empire and spreading). *Assume a proof structure π, an eigen weight p and a weight w, then:*

- *the* dependency graph *of p (w.r.t. π), denoted $\mathcal{E}_p$, is the (possibly disconnected) subgraph of π made by all links depending on p;*
- *the* spreading *of w over π, denoted by $w.[\pi]$, is the product of w for π, i.e., π where we replaced each weight v with the product of weights vw.*

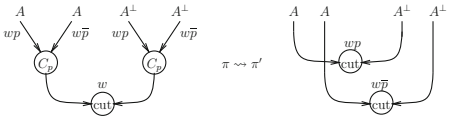
Definition 10 (commutative cut reduction). *Let L be cut link in a proof net π whose premises A and $A^\perp$ are the respective conclusions of links L' and L'' s. t. at least one of them is a contraction link C. Then we define the result π' (*reductum*) of reducing this* commutative *cut L in π (*redex*), as follows:*

*$(C/\parr)$***-cut:** *if L' is a C-link and L'' is a $\parr$-link, then π reduces in one $(C/\parr)$ step to π' (the (C/∇) step is analogous) as follows:*

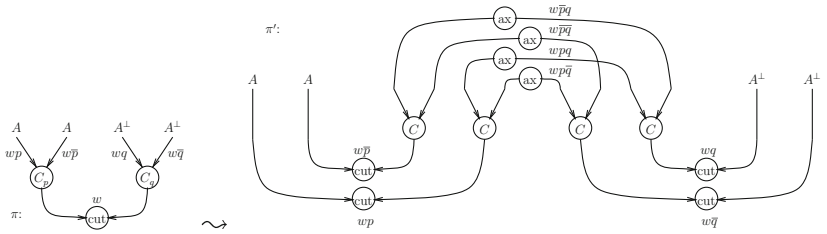


*(C/C)***-cut:** *if both L' and L'' are C-links, then there are two cases:*

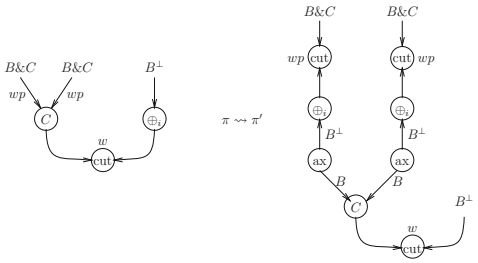
1. *either the weight w of both L' and L'' splits on the same p variable, then π reduces in one (C_p/C_p) step to π' as follows*



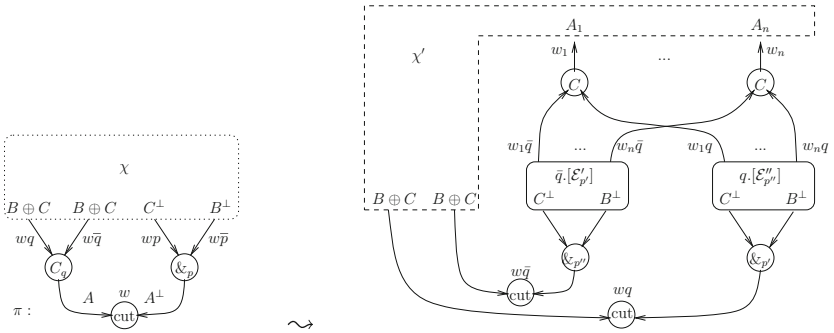
2. *or the weight w of L' (resp., w of L'') splits on p (resp., on q) then π reduces in one (C_p/C_q) step to π' as follows*



$(C/\oplus_i)$**-cut:** *if L' is a C-link and L'' a $\oplus_{i=1,2}$-link, then π reduces in one $(C/\oplus)$ step to π' as follows*



$(C/\&)$**-cut:** *if L' is a C-link and L'' a $\&_p$-link, then π reduces in one $(C/\&)$ step to π' as follows*



with the assumptions that graphs $\bar{q}.[\mathcal{E}'_{p'}]$ and $q.[\mathcal{E}''_{p''}]$ are obtained as follows:

1. *we take two copies, $\mathcal{E}'_p$ and $\mathcal{E}''_p$, of the dependency graph $\mathcal{E}_p$ of p;*
2. *we replace in $\mathcal{E}'_p$ (resp., in $\mathcal{E}''_p$) p with a new variable p' (resp., p'');*
3. *we spread $\bar{q}$ (resp., q) over $\mathcal{E}'_{p'}$ (resp., over $\mathcal{E}''_{p''}$).*

Technical details of proofs of Theorems 2, 3 and 4 can be found in [14].

Theorem 2 (stability). *If π is a GPN that reduces to π' in one step of commutative cut reduction then π' is a GPN too.*

Theorem 3 (termination). *We can always reduces a proof net π into a proof net π' that is cut-free, by iterating the reduction steps of Definitions 8 and 10.*

Theorem 4 (confluence). *Assume π is a proof net s.t. it reduces in one step α to π' ($\pi \rightsquigarrow_\alpha \pi'$) and it reduces in an other step β to π'' ($\pi \rightsquigarrow_\beta \pi''$); then, there exists a proof net σ such that both π' reduces, in a certain number of steps, to σ ($\pi' \rightsquigarrow^* \sigma$) and π'' reduces, in a certain number of steps, to σ ($\pi'' \rightsquigarrow^* \sigma$).*

2.3 Sequentialization

There exists a correspondence, called *sequentialization* (Theorem 5), between PNs and sequential proofs.

Lemma 1 (splitting). *Let π be a CyMLL PN with at least a $\otimes$-link or cut-link and with conclusions Γ not containing any terminal ∇-link (so, we say π is in* splitting condition*); then, there must exist a $\otimes$-link $\frac{A \quad B}{A \otimes B}$ (resp., a cut-link $\frac{A \quad A^{\perp}}{}$) that splits π in two CyMLL PNs, π_A and π_B (resp., π_A and $\pi_{A^{\perp}}$).*

Proof. Consequence of the Splitting Lemma for commutative MLL PNs [7].

Lemma 2 (PN cyclic order conclusions). *Let π be a CyMLL PN with conclusions Γ, then all seaweeds $S_i(\pi) \downarrow^{\Gamma}$, restricted to Γ, induce the same cyclic order σ on Γ, denoted $\sigma(\Gamma)$ and called* the (cyclic) order of the conclusions of π.

Proof. By induction on the size $\langle \sharp V, \sharp E \rangle$[2] of π.

Corollary 1 (stability of PN order conclusions under cut reduction). *If π, with conclusions $\sigma(\Gamma)$, reduces in one step of cut reduction to π', then also π' has conclusions $\sigma(\Gamma)$.*

Theorem 5 (sequentialization of CyMLL PNs). *Any CyMLL PN with conclusions $\sigma(\Gamma)$ can be* sequentialized *into a CyMLL sequent proof with the same cyclic order conclusions $\sigma(\Gamma)$.*

Proof. By induction on the size of the given proof net π via Lemmas 1 and 2.

Theorem 6 (sequentialization of CyMALL PNs). *A CyMALL GPN with conclusions $\sigma(\Gamma)$ can be* sequentialized *into a CyMALL sequent proof with the same cyclic conclusions $\sigma(\Gamma)$ and vice-versa (*de-sequentialization*).*

Proof. There are two parts.

Sequentialization-part. Any CyMALL proof net π can be sequentialized into a proof Π, by induction on the number of &-links. The base of induction corresponds to the sequentialization of the CyMLL proof nets (Theorem 5). The induction step follows by the sequentialization of standard MALL PNs (see [8]) where the only novelty is to show that: if a PN π contains a terminal &-link L, then π can be *toggled*[3] at L in two PNs preserving conditions 1 and 2 of Definition 7.

De-sequentialization-part. Any CyMALL proof Π of $\sigma(\Gamma)$ can be de-sequentialized into a PN π of $\sigma(\Gamma)$, by induction of the height of Π derivation.

[2] Number of vertexes and number of edges.

[3] We say that a terminal $\&_p$-link of a GPN π is *toggling* when the restriction of π w.r.t. p and the restriction of π w.r.t. $\bar{p}$ are both correct GPSs. We call the *restriction of π w.r.t. p* (resp., *w..r.t. $\bar{p}$*) what remains of π when we replace p with 1 (resp., $\bar{p}$ with 1) and keep only those vertexes and edges whose weights are still non-zero (see [8]).

Unlike most part of correctness criteria for non-commutative proof nets, like [3,13], our syntax enjoys a sequentialization for the full class of CyMLL PNs (with possible cuts). Observe that, *Mellies's counter-example* (Example 1) represents a non-sequentializable proof structure that becomes correct (therefore, sequentializable) after cut reduction.

3 Embedding Lambek Calculus into CyMALL PNs

Definition 11 (Lambek formulas and sequents of CyMALL). *Assume A and S are, respectively, a formula and a sequent of CyMALL.*

1. *A is a* pure Lambek formula (pLF) *if it is a CyMLL formula recursively built according to this grammar: $A := \text{positive atoms} \mid A \otimes A \mid A^{\perp} \triangledown A \mid A \triangledown A^{\perp}$.*
2. *A is an* additive Lambek formula (aLF or simply LF) *if it is a CyMALL formula recursively built according this grammar: $A := pLF \mid A \& A \mid A \oplus A$.*
3. *S is a* Lambek sequent of CyMALL *iff $S = (\Gamma^{\perp}, A)$, where A is a non-void LF and $\Gamma^{\perp}$ is a possibly empty finite sequence of negations of LFs (i.e., Γ is a possibly empty sequence of LFs and $\Gamma^{\perp}$ is obtained by taking the negation of each formula in Γ).*
4. *A* Lambek proof *is any derivation built by means of the CyMALL inference rules whose premise(s) and conclusions are CyMALL Lambek sequents.*

Definition 12 (Lambek proof net). *We call* Lambek CyMALL proof net *(resp.,* pure Lambek CyMLL proof net*) any CyMALL PN (resp., CyMLL PN) whose edges are labeled by LFs (resp. pLFs) or negation of LFs (resp., pLFs) and whose conclusions form a Lambek sequent.*

Corollary 2. *Any Lambek CyMALL proof net π is stable under cut reduction, i.e., if π reduces in one step to π', then π' is a Lambek CyMALL proof net too.*

Proof. Consequence of Theorems 1 and 2. Trivially, each reduction step preserves the property that each edge of the reductum is labeled by a Lambek formula or the negation of a Lambek formula.

Theorem 7 (de-sequentialization of Lambek CyMALL proofs). *Any proof of a CyMALL Lambek sequent $\vdash \sigma(\Gamma^{\perp}, A)$ can be de-sequentialized into a Lambek CyMALL PN with conclusions $\sigma(\Gamma^{\perp}, A)$.*

Proof. By induction on the height of the given sequent proof (similarly to the de-sequentialization part of Theorem 6).

Theorem 8 (sequentialization of Lambek CyMALL PNs). *Any Lambek CyMALL PN of $\sigma(\Gamma^{\perp}, A)$ can be sequentialized into a Lambek CyMALL proof of the sequent $\vdash \sigma(\Gamma^{\perp}, A)$.*

Proof. Sequentialization follows by induction on the number &-links of the given PN. The base of induction is given by next Theorem 9. The induction step, simply follows by Theorem 6.

Theorem 9 (sequentialization of pure Lambek CyMLL PNs). *Any Lambek CyMLL PN of* $\sigma(\Gamma^{\perp}, A)$ *can be sequentialized into a Lambek CyMLL proof of* $\vdash \sigma(\Gamma^{\perp}, A)$.

Proof. See details in [2].

4 Language Parsing with Lambek CyMALL PNs

In order to show how powerful PNs are, in this section we adapt to our syntax, some linguistics (typing) examples suggested by Richard Moot in his PhD thesis [18]. We use s, np and n as the types expressing, respectively, a *sentence*, a *noun phrase* and a *common noun*. According to the "parsing as deduction style", when a string $w_1...w_n$ is tested for grammaticality, the types $t_1, ..., t_n$ associated with the words are retrieved from the lexicon and then parsing reduces to proving the derivability of a two-sided sequent of the form $t_1, ..., t_n \vdash s$. Recall that proving a two sided Lambek derivation $t_1, ..., t_n \vdash s$ is equivalent to prove the one-sided sequent $\vdash t_n^{\perp}, ...t_1^{\perp}, s$ where $t_i^{\perp}$ is the dual (i.e., the linear negation) of each type t_i. Therefore, any phrase or sentence should be written like in a mirror (observing the opposite natural direction).

Assume the following lexicon, where *linear implication* $\multimap$ (resp., $\circ\!\!-$) is traditionally used for expressing types in two-sided sequent parsing style:

1. Vito $= np$;
2. Sollozzo $= np$;
3. him $= (s \circ\!\!- np) \multimap s \;=\; (s \nabla np^{\perp})^{\perp} \nabla s \;=\; (np \parr s^{\perp}) \nabla s$;
4. trusts $= (np \multimap s) \circ\!\!- np \;=\; (np^{\perp} \nabla s) \nabla np^{\perp}$.

Cases of lexical ambiguity follow to words with several possible formulas A and B assigned it. For example, a verb like "*to believe*" can express a relation between two persons, np's in our interpretation, or between a person and a statement, interpreted as s, like in these examples:

(1) *Sollozzo believes Vito*; (2) *Sollozzo believes Vito trusts him.*

We can express this polymorphism by two lexical assignments as follows:

5. believes $= (np \multimap s) \circ\!\!- np \;=\; (np^{\perp} \nabla s) \nabla np^{\perp}$;
6. believes $= (np \multimap s) \circ\!\!- s \;=\; (np^{\perp} \nabla s) \nabla s^{\perp}$.

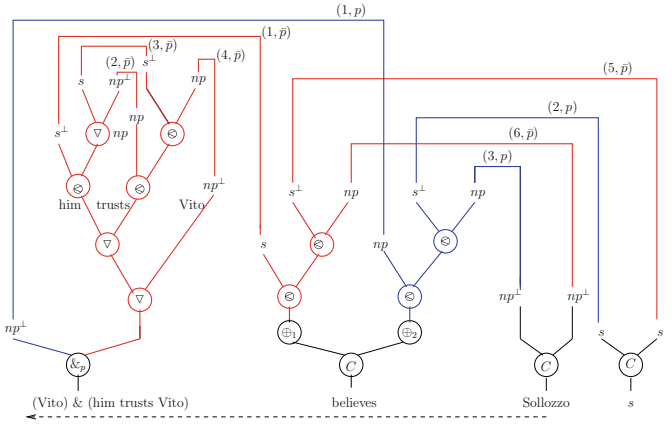
Typically, additives are used to capture cases of lexical ambiguity. When a word has two possible formulas A and B assigned it, we can combine these into a single additive formula $A \& B$ (resp., $A \oplus B$). Thus, we can collapse assignments 5 and 6 into the following single additive assignment:

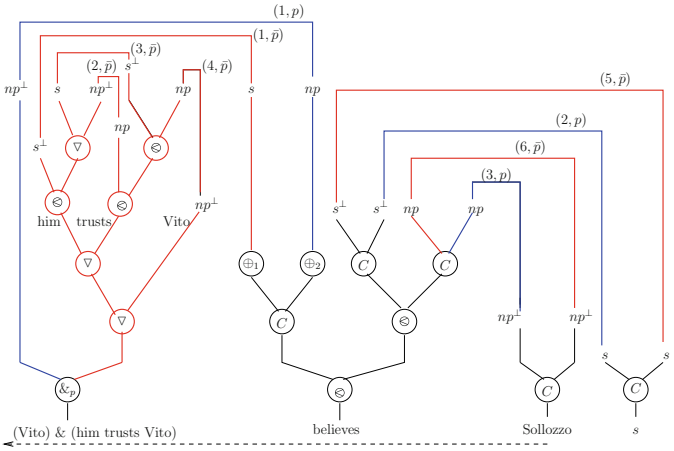
7. believes $= ((np \multimap s) \circ\!\!- np) \& ((np \multimap s) \circ\!\!- s) \;=\; ((np^{\perp} \nabla s) \nabla np^{\perp}) \& ((np^{\perp} \nabla s) \nabla s^{\perp})$.

Equivalently, via distributivity of negative connectives, we could also move the additive "inside" and generate a more compact lexical entry, in which the two assignments share their identical initial parts (see also [19] on type polymorphism):

8. believes $= ((np \multimap s) \circ\!\!- (s \oplus nps)) = ((np^\perp \triangledown s) \triangledown (np^\perp \& s^\perp))$.

Using that, we can then move lexical ambiguity into proof nets. In the following we give two equivalent Lambek PNs as parsing of the additive superposition of sentences (1) and (2); the first (resp., the second) PN makes use of the lexical entry 7 (resp., the lexical entry 8).





Observe that, for each slice of each proof net above, $\varphi_p(\pi)$ and $\varphi_{\bar{p}}(\pi)$, there exists an additive resolution that is a CyMLL PN with the same sequentialization:

$$\varphi_p(\pi) \;\Rightarrow\; \Pi_1 : \quad \dfrac{\dfrac{}{np^\perp, np}\, ax_1^p \qquad \dfrac{\dfrac{}{s^\perp, s}\, ax_2^p \qquad \dfrac{}{np, np^\perp}\, ax_3^p}{s^\perp \otimes np, np^\perp, s}\,\otimes}{np^\perp, np \otimes (s^\perp \otimes np), np^\perp, s}\,\otimes$$

$$\varphi_{\bar{p}}(\pi) \;\Rightarrow\; \Pi_2 : \quad \dfrac{\dfrac{\dfrac{\dfrac{}{s^\perp, s}\, ax_1^{\bar{p}} \qquad \dfrac{\dfrac{s, np^\perp, np \otimes (s^\perp \otimes np), np^\perp}{s \nabla np^\perp, np \otimes (s^\perp \otimes np), np^\perp}\,\nabla}{s^\perp \otimes (s \nabla np^\perp), np \otimes (s^\perp \otimes np), np^\perp, s}\,\otimes}{\dfrac{s^\perp \otimes (s \nabla np^\perp), np \otimes (s^\perp \otimes np), np^\perp, s \otimes (s^\perp \otimes np), np^\perp, s}{((s^\perp \otimes (s \nabla np^\perp)) \nabla (np \otimes (s^\perp \otimes np))) \nabla np^\perp, s \otimes (s^\perp \otimes np), np^\perp, s}\,\nabla^2} \qquad \dfrac{\dfrac{}{s^\perp, s}\, ax_5^{\bar{p}} \qquad \dfrac{}{np, np^\perp}\, ax_6^{\bar{p}}}{s^\perp \otimes np, np^\perp, s}\,\otimes}{}\,\otimes \qquad \dfrac{\dfrac{}{np^\perp, np}\, ax_2^{\bar{p}} \qquad \dfrac{\dfrac{}{s^\perp, s}\, ax_3^{\bar{p}} \qquad \dfrac{}{np, np^\perp}\, ax_4^{\bar{p}}}{s^\perp \otimes np, np^\perp, s}\,\otimes}{}\,\otimes$$

5 Conclusions

As future work we aim at investigating the how topological correctness criteria based on graph rewriting (or retraction) of MALL proof structures [6,15] may be used for (linguistic) parsing. Naively, retraction criteria allow to switch from the paradigm of "parsing as deduction" to the paradigm of "parsing as rewriting" (see, e.g., [16]). Moreover, retraction could also be a useful computational tool for studying the complexity class of the CyMALL correctness criterion.

Acknowledgements. We thank the anonymous reviewers, Michael Moortgat and Richard Moot for their useful comments and suggestions. This work was partially supported by the PRIN Project *Logical Methods of Information Management*.

References

1. Abrusci, V.M.: Classical conservative extensions of Lambek calculus. Stud. Logica. **71**(3), 277–314 (2002)
2. Abrusci, V.M., Maieli, R.: Cyclic multiplicative proof nets of linear logic with an application to language parsing. In: de Paiva, V., de Queiroz, R., Moss, L.S., Leivant, D., de Oliveira, A. (eds.) WoLLIC 2015. LNCS, vol. 9160, pp. 53–68. Springer, Heidelberg (2015)
3. Abrusci, V.M., Ruet, P.: Non-commutative logic I: the multiplicative fragment. Ann. Pure Appl. Logic **101**(1), 29–64 (2000)
4. Andreoli, J.-M., Pareschi, R.: From Lambek calculus to word-based parsing. In: Proceedings of Substructural Logic and Categorial Grammar Workshop, Munchen (1991)
5. Danos, V., Regnier, L.: The structure of multiplicatives. Arch. Math. Logic **28**, 181–203 (1989)
6. Danos, V.: La Logique Linéaire appliquée à l'étude de divers processus de normalisation (principalment du λ-calcul). Ph.D. thesis, Paris (1990)
7. Girard, J.-Y.: Linear logic. Theoret. Comput. Sci. **50**, 1–102 (1987)
8. Girard, J.-Y.: Proof-nets: the parallel syntax for proof theory. In: Logic and Algebra. Marcel Dekker (1996)
9. Girard, J.-Y.: Le point aveugle. Cours de Logique, vol. I, Vers la Perfection. Ed. Hermann, Paris (2006)
10. Hughes, D., van Glabbeek, R.: Proof nets for unit-free multiplicative-additive linear logic. In: Proceedings of IEEE LICS (2003)
11. Lambek, J.: The mathematics of sentence structure. Amer. Math. Monthly **65**, 154–170 (1958)

12. Laurent, L., Maieli, R.: Cut elimination for monomial MALL proof nets. In: Proceedings of IEEE LICS, Pittsburgh, USA, pp 486–497 (2008)
13. Maieli, R.: A new correctness criterion for multiplicative non-commutative proof-nets. Arch. Math. Logic **42**, 205–220 (2003). Springer-Verlag
14. Maieli, R.: Cut elimination for monomial proof nets of the purely multiplicative and additive fragment of linear logic. IAC-CNR Report, no. 140 (2/2008). HAL Id: hal-01153910. https://hal.archives-ouvertes.fr/hal-01153910
15. Maieli, R.: Retractile proof nets of the purely multiplicative and additive fragment of linear logic. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007. LNCS (LNAI), vol. 4790, pp. 363–377. Springer, Heidelberg (2007)
16. Maieli, R.: Construction of retractile proof structures. In: Dowek, G. (ed.) RTA-TLCA 2014. LNCS, vol. 8560, pp. 319–333. Springer, Heidelberg (2014)
17. Moot, R., Retoré, C.: A logic for categorial grammars: Lambek's syntactic calculus. In: Moot, R., Retoré, C. (eds.) The Logic of Categorial Grammars. LNCS, vol. 6850, pp. 23–63. Springer, Heidelberg (2012)
18. Moot, R.: Proof nets for linguistic analysis. Ph.D. thesis. Utrecht University (2002)
19. Morrill, G.: Additive operators for polymorphism. In: Categorial Grammar: Logical Syntax, Semantics and Processing. Oxford University Press (2011)