The Exponential Logic of Sequentialization

Aurore Alcolei[a,1]   Luc Pellissier[b,2]   Alexis Saurin[c,3]

[a] *Université Paris Est Creteil, LACL, F-94010 Créteil, France*

[b] *Université Paris Est Creteil, LACL, F-94010 Créteil, France*

[c] *IRIF, CNRS, Université Paris Cité & INRIA, Paris, France*

Abstract

Linear logic has provided new perspectives on proof-theory, denotational semantics and the study of programming languages. One of its main successes are proof-nets, canonical representations of proofs that lie at the intersection between logic and graph theory. In the case of the minimalist proof-system of multiplicative linear logic without units (MLL), these two aspects are completely fused: proof-nets for this system are graphs satisfying a correctness criterion that can be fully expressed in the language of graphs.

For more expressive logical systems (containing logical constants, quantifiers and exponential modalities), this is not completely the case. The purely graphical approach of proof-nets deprives them of any sequential structure that is crucial to represent the order in which arguments are presented, which is necessary for these extensions. Rebuilding this order of presentation — sequentializing the graph — is thus a requirement for a graph to be logical. Presentations and study of the artifacts ensuring that sequentialization can be done, such as boxes or jumps, are an integral part of researches on linear logic.

Jumps, extensively studied by Faggian and di Giamberardino, can express intermediate degrees of sequentialization between a sequent calculus proof and a fully desequentialized proof-net. We propose to analyze the logical strength of jumps by internalizing them in an extention of MLL where axioms on a specific formula, the jumping formula, introduce constrains on the possible sequentializations. The jumping formula needs to be treated non-linearly, which we do either axiomatically, or by embedding it in a very controlled fragment of multiplicative-exponential linear logic, uncovering the exponential logic of sequentialization.

Keywords: jumps, linear logic, proof theory, proof nets, sequentialization

1   Introduction

Proof theory is concerned with proof systems, specifying how proofs are structured, and in turn, which formulæ are provable. Different proof systems have different properties, such as soundness (the fact that all provable formulæ are also valid — for a given semantics of formulæ), completeness (the fact that all valid formulæ are also provable — again, for a given semantics of formulæ), canonicity (two proofs differ if and only if they have different interpretations — for a given semantics of proofs), ...

Broadly speaking, Linear logic [11] has two main kinds of proof systems:

---

[1]   aurore.alcolei@ens-lyon.org

[2]   luc.pellissier@u-pec.fr

[3]   alexis.saurin@irif.fr

Sequent calculus Proofs are trees whose nodes are labelled by inference rules transforming sequents of formulæ. This representation – even though it is both historically decisive in the development of proof-theory since the work of Gentzen by emphasizing the role of the cut inference as well as for reductive logics (aka. proof-construction) by suggesting a natural notion of proof-goal – contains lot of unnecessary information (such as the specific order of application of some unrelated rules): it is far from being canonical. The loss of confluence in sequent calculus witnesses this loss of canonicity.

Proof-nets are graphs whose nodes relate formulæ to one another. A proof-net contains much less information than a sequent calculus proof with respect to the order of application of inference rules. This allows them to be canonical, for some fragments of linear logic at least. In particular, they are canonical for multiplicative linear logic without units (simply referred to as MLL in the following). As a by-product, confluence is recovered in MLL proof-nets.

Apart from the contrast on canonicity issue, a major difference between sequent proofs and proof-nets follows from the tree *vs.* graph structure and concerns logical correctness of the proof-object: sequent proofs being inductively defined proof objects, their logical correctness is naturally defined in a *local* way: soundness of the conclusion follows from the soundness of each premise and one only considers derivation trees that is built applying inferences in a correct way. On the other hand, proof-nets being graphs, there is no such thing as a root, in general, and logical correctness therefore becomes a *global* property of the graph. One of this property is *saturation*, the fact that the graph can be saturated with more edges to produce a proof tree, called a *sequentialization* of the net. It would be ad hoc to restrict solely to those graphs which are logically correct and one therefore consider two levels of proof objects: **proof structures**, which are graphs with some typing constraints on the edges and vertices; and **proof-nets**, which are logically correct proof structures.

For instance, consider the following rules of linear logic sequent calculus:

$$\frac{}{\vdash \mathbf{1}} \; \mathbf{1} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \bot} \; \bot$$

They introduce the so-called multiplicative constants $\mathbf{1}$ and $\bot$. The way they are introduced differ on their context: while $\mathbf{1}$ can be deduced thanks to an axiom, in an empty content, $\bot$ needs to be juxtaposed to an already derived context. So, in a proof-net system, where operations are not on sequents of formulæ but on formulæ, this contextuality is missing. Indeed, both introductions are presented as a lone node with no input and an output:

Context has thus to be recovered. One way to do so is via correctness criterion, an extra *property* verified by the graph, ensuring that there exists a coherent way of assigning its context to each formula. Another way is to add extra *structure* on top of the graph. The most common such structure is the *box*, introduced in Girard's seminal paper [11]. Consider the sequent calculus rule introducing the exponential modality !:

$$\frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A} \; !$$

For a formula to receive a !, all the other formulæ in the context have to be under the ? modality. A way to translate that into proof-nets is by requiring that a special region of the graph is enclosed so that every formula going out of it has a ?, except one corresponding to the !-cell. This is a direct translation of the sequent calculus (where rules are applied to sequents and not formulæ) into proof-nets, in a graph-theoretically awkward fashion:
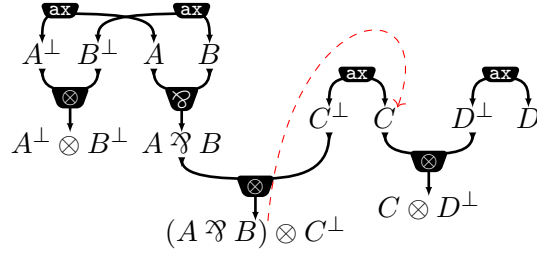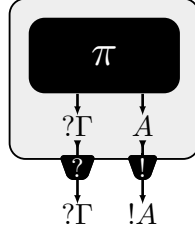
Fig. 1. A graphical jump

Another more graph-theoretical way to introduce contexts are *jumps*, a recurrent feature in the literature on proof nets [13].

They are purely geometrical (i.e. untyped) edges adding extra connection between links of a proof structure. Jumps were first introduced by Girard in [13] to add more sequentiality in proof nets with quantifiers, typically preventing sequentialization that would not match the variable-witness dependencies between $\forall$ and $\exists$ quantifier. Such jumps can also be found in unification nets, a more recent framework of proof nets with quantifiers that bears implicit witnesses [15]. Jumps are also used as lighter feature than boxes to handle the sequentiality induced by sequent rules outside of MLL, typically the unit rules [4,14,17], additive rules [14,16] and exponentials for λ-nets [1].

On another line of work, Faggian and her collaborators developed a setting in which jumps are taken seriously as edges in a more general graph than just the mere syntactic tree of a formula. First in the context of *L-nets* [5,6,10], a parallel syntax for Ludics designs, then in the context of *J-proof nets* for HS$^+$ (a polarized and hypersequentialized sequent calculus) and MLL [8,9], their setting allows in particular to see sequent calculus proofs as proof-nets saturated with sequential edges, embedding both proof-nets and sequent calculus proofs in a same universe of partially sequentialized proof-structures, each of them living at one extremity on a de·sequentialization spectrum.

The figure above depicts a proof net with a jump (the red dashed arrow) from (the formula occurrence) $(A \,\mathfrak{P}\, B) \otimes C^\perp$ to (the formula occurrence) C. As a consequence, when sequentializing such a net, the $\otimes$ rule introducing $(A \,\mathfrak{P}\, B) \otimes C^\perp$ will always be placed above the $\otimes$ rule using C to introduce $C \otimes D^\perp$. Jumps are thus a way to *constrain* the set of sequentializations that are derivable from a proof net.

Proof structures with jumps offer a larger setting than usual proof structures, it is thus natural to search how the behave under cut elimination and how the usual correctness criteria on proof nets can be adapted to proof nets with jumps.

Contributions and organization of the paper.

In this paper, we show that graphical jumps as presented above can be internalised in the logic under study at the only cost of adding a special atom **J** that enjoys, as well as its dual, the property of a monoid and behaves as an exponential under cut elimination. This atom can itself be encoded in a fragment of linear logic, thus giving a way to find correctness criteria for proof structure with jumps through the ones of usual proof structures.

To keep this paper light and readable, we will focus our internalisation of jumps to MLL. In Section 2, we will recast the basic definitions on MLL sequent proofs and proof-nets. In Section 3 we start introducing our encoding of jumps as axioms in a cut-free setting. In particular, we will define MLL$_\mathbf{J}$ a logic which is basically MLL plus a special atom **J** that enjoys left and right contractions rules. We will prove that our

$$\dfrac{\dfrac{}{\vdash A, A^{\perp}}\ \text{Ax} \qquad \dfrac{\pi}{\vdash A, \Delta}}{\vdash A, \Delta}\ \text{cut} \qquad \rightarrow \dfrac{\pi}{\vdash A, \Delta}$$

$$\dfrac{\dfrac{\dfrac{\theta}{\vdash \Theta, F, G}}{\vdash \Theta, F \,\mathbin{\rotatebox[origin=c]{180}{\&}}\, G}\ \mathbin{\rotatebox[origin=c]{180}{\&}} \qquad \dfrac{\dfrac{\gamma}{\vdash \Gamma, F^{\perp}} \quad \dfrac{\delta}{\vdash \Delta, G^{\perp}}}{\vdash \Gamma, \Delta, F^{\perp} \otimes G^{\perp}}\ \otimes}{\vdash \Theta, \Gamma, \Delta}\ \text{cut}$$

$$\rightarrow \dfrac{\dfrac{\dfrac{\theta}{\vdash \Theta, F, G} \quad \dfrac{\gamma}{\vdash \Gamma, F^{\perp}}}{\vdash \Theta, G, \Gamma}\ \text{cut} \quad \dfrac{\delta}{\vdash \Delta, G^{\perp}}}{\vdash \Theta, \Gamma, \Delta}\ \text{cut}$$

$$\text{or} \quad \dfrac{\dfrac{\dfrac{\theta}{\vdash \Theta, F, G} \quad \dfrac{\delta}{\vdash \Delta, G^{\perp}}}{\vdash \Theta, F, \Delta}\ \text{cut} \quad \dfrac{\gamma}{\vdash \Gamma, F^{\perp}}}{\vdash \Theta, \Gamma, \Delta}\ \text{cut}$$

Fig. 2. Sequent calculus reductions

encoding is correct and that our jumps correspond exactly to the ones described in [9], in the sense that their correctness criterion matches with the usual Danos-Regnier criterion on our encoding. In Section 4 we will discuss the encoding of jumps in the dynamic setting of proofs with cuts and cut elimination, enriching MLL$_\mathbf{J}$ with the appropriate exponential dynamics for $\mathbf{J}$. This will make explicit the dynamics of jumps sketched in [9] and treated more exhaustively in [7]. Finally, we will conclude in Section 5 by showing that the jump atom $\mathbf{J}$ can be implemented effectively in the exponential fragment of MELL + Mix.

2   MLL proofs: sequent-calculus, proof-nets and sequentialization

MLL is a minimal proof-system. Its formulæ are given by the following grammar (with A ranging in a set of atomic formulæ):

$$F = F \otimes F \mid F \,\mathbin{\rotatebox[origin=c]{180}{\&}}\, F \mid A \mid A^{\perp}.$$

Linear negation is inductively defined by

$$(A^{\perp})^{\perp} = A \qquad (F \otimes G)^{\perp} = G^{\perp} \,\mathbin{\rotatebox[origin=c]{180}{\&}}\, F^{\perp} \qquad (F \,\mathbin{\rotatebox[origin=c]{180}{\&}}\, G)^{\perp} = G^{\perp} \otimes F^{\perp}.$$

Sequent calculus for MLL

MLL proofs, in the sequent calculus, are given by the applications of the rules:

$$\dfrac{}{\vdash A, A^{\perp}}\ \text{Ax} \qquad \dfrac{\vdash \Gamma, F, G}{\vdash \Gamma, F \,\mathbin{\rotatebox[origin=c]{180}{\&}}\, G}\ \mathbin{\rotatebox[origin=c]{180}{\&}} \qquad \dfrac{\vdash \Gamma, F \quad \vdash \Delta, G}{\vdash \Gamma, \Delta, F \otimes G}\ \otimes \qquad \dfrac{\vdash \Gamma, F \quad \vdash \Delta, F^{\perp}}{\vdash \Gamma, \Delta}\ \text{cut}$$

Its dynamic of reduction, also called *cut elimination* is recalled in figure 2. The resulting calculus is (non deterministic and) not confluent.

Remark 2.1 We have not made precise what sequents are, in the previous definition. There are many possible choices which have various impacts. We leave this as implicit and lightweight as possible in the remaining of the paper but need to discuss this here. We crucially need a notion of sequents which allows for two properties: (i) to trace *formula occurrences* along the proof, that is admit a well-defined notion of formula ancestor (ii) which validates the exchange rule (as an admissible rule at least).

Among the standard options at hand, there are sequents as ordered lists of formulas, sequents as sets of locative formula occurrences or finally sequents as indexed sets of formulas. On the other hand, sequents as multisets of formulas are not an option as it does not allow for a well-defined notion of formula occurrence,
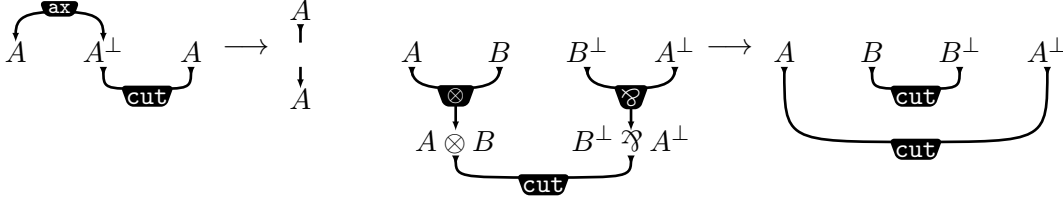
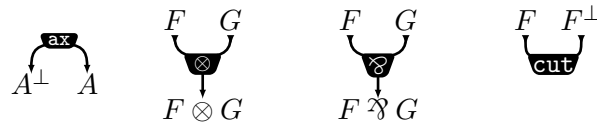Fig. 3. Proof-structure reductions

nor a well-defined notion of desequentialization (see below). The first option requires to consider some form of an exchange rule: sequents are lists of formulas, written $\vdash F_1, \ldots, F_n$, and the following exchange rule is necessary: $\dfrac{\vdash \Gamma, G, F, \Delta}{\vdash \Gamma, F, G, \Delta}$ ex . In such a situation, one can treat, as often, the exchange rule implicitly by considering that the set of inferences is complemented with the derived rules obtained by pre- and post-composing every other rule with a series of exchanges. (See Example 2.2 for an illustration, if sequents are viewed as ordered lists there.) Each such derived rule is equipped with a notion of ancestor derived from the basic inference rules given above (as usually done in the proof theory literature [3]).

When sequents are treated in a locative way, their formulas are equipped with a unique address (pairwise incomparable) and the notion of sub-address provides an explicit notion of formula ancestor, the exchange rule in unnecessary in this case. (See [2,12] for instance.) Last, with sequents as indexed sets of formulas, each inference rules is equipped with a relation between the indexing set of the conclusion of the rule and that of its premises, constituting the ancestor relation. (Note that the above two presentations are concrete instances of this last presentation: with lists the indexing sets are the positions in the list, with locations, the indexing sets are finite sets of pairwise incomparable addresses.)

In the following, we shall forget about the above considerations and simply assume that we have a well-defined notion of formula occurrence and formula ancestor.

Proof structures for MLL

Among sequent calculus proofs of a given sequent, some are essentially different, while some are essentially the same: a difference in the order of introduction of two connectives is typically a non-essential difference. A canonical representation of proofs can be found when moving to the highly parallel setting of proof-nets, a graphical representation of proofs. A MLL *proof-structure* is a directed graph whose nodes (links) and edges abide to the following typing constraints:

Note that the type of the edges of a proof-structure can be recovered from the type of the edges going out its axiom links and the type of its links, so that they could be made implicit although they are often kept for readability reasons. Besides, we say that such a graph is a *proof-structure for the sequent* Γ if we obtain Γ by collecting all the formulas of its pending edges up to reordering.

Reflecting the dynamics of sequent calculus proofs, proof structures also define a (non-deterministic) graph rewriting calculus; the corresponding reductions are recalled in Figure 3. Cut elimination in proof-structures is non-deterministic but *confluent* and *terminating*. We write $\Downarrow R$ for the (unique) normal form of a proof-structure R; $\Downarrow_\rho R$ if we want to make more precise the sequence of reduction steps leading to it [4] .

---

[4] A sequence of reduction steps is simply given as a sequence of pairs of the activated cut-formulas – remember that in MLL proof nets there is no commutation rule.

Sequent calculus proofs as (constrained) proof structures

Let us consider an MLL proof π. A formula F is a *formula occurrence* in π if it is the active conclusion of a rule in π (the formula having its topmost connective introduced by the rule or the two formulas introduced by an axiom rule). For F a formula occurrence in π, we define its *life-time* as sub-branch from its creation to its use.

For two rule occurrences r and r' in a proof π, we say that r is *above* (or *occurs before*) r' if r and r' belong to the same branch in the proof, and r' appears before r starting from the root. If r produces a formula F this is equivalent to saying that there exists F', an active formula of r', such that F is created before F' is used; in that case, we write $F \prec F'$.
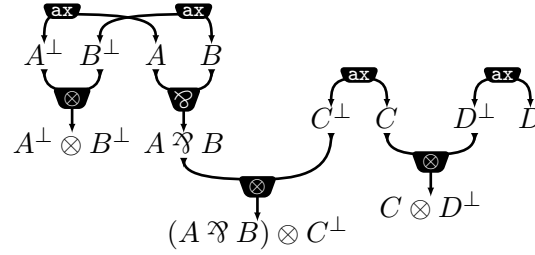
Every sequent proof π can be desequentialized into a proof structure R_π, defined inductively. For a proof structure R, we note $\mathsf{seq}(R)$ its set of sequentializations, that is, the set of sequent proofs π such that $R_\pi = R$. Desequentialization is preserved by cut elimination and, conversely, if $\pi \in \mathsf{seq}(R)$ and $R \to R'$, then there exists $\pi' \in \mathsf{seq}(R')$ such that $\pi \to^+ \pi'$. Note, however, that in general proofs in $\mathsf{seq}(R')$ do not correspond to the reduct of some proof in $\mathsf{seq}(R)$.

If π is a sequentialization of R then there is a one-to-one correspondence between the rules in π and the links in R (this correspondence is defined during the inductive procedure of desequentialization). By extension every edge in R can be associated with a formula occurrence in π throughout its life-time. In the following we will often make use of this correspondence, keeping the conversion from proof to net implicit.

Given a proof structure R, its set of sequentializations may be empty: in that case we say that R is *incorrect*. On the contrary, if $\mathsf{seq}(R)$ is non empty, we say that R is *correct* or that it is *a proof net*, meaning that it actually corresponds to a canonical representation of proofs.

A proof net can have several sequentializations, for example:

Example 2.2 Consider the (correct) proof-structure R:

It has two different sequentializations:

$$
\cfrac{
  \cfrac{
    \cfrac{
      \overline{\vdash A^\perp, A} \quad \overline{\vdash B^\perp, B}
    }{\vdash A^\perp \otimes B^\perp, A, B} \otimes
  }{\vdash A^\perp \otimes B^\perp, A \,\mathfrak{P}\, B} \,\mathfrak{P} \quad \overline{\vdash C^\perp, C}
}{\cfrac{\vdash A^\perp \otimes B^\perp, (A \,\mathfrak{P}\, B) \otimes C^\perp, C}{\vdash A^\perp \otimes B^\perp, (A \,\mathfrak{P}\, B) \otimes C^\perp, C \otimes D^\perp, D} \otimes \quad \overline{\vdash D^\perp, D}} \otimes
$$

$$
\cfrac{
  \cfrac{
    \cfrac{
      \overline{\vdash A^\perp, A} \quad \overline{\vdash B^\perp, B}
    }{\vdash A^\perp \otimes B^\perp, A, B} \otimes
  }{\vdash A^\perp \otimes B^\perp, A \,\mathfrak{P}\, B} \,\mathfrak{P} \quad
  \cfrac{\overline{\vdash C^\perp, C} \quad \overline{\vdash D^\perp, D}}{\vdash C^\perp, C \otimes D^\perp, D} \otimes
}{\vdash A^\perp \otimes B^\perp, (A \,\mathfrak{P}\, B) \otimes C^\perp, C \otimes D^\perp, D} \otimes
$$

On the first one, A is above D, not in the second.

Graphical jumps as depicted in Figure 1 are used to restrict the possible set of sequentializations of a net: if there is a jump from F to F' in R, then we expect F to appear above F' in any sequentialization of R. Writing jumps as a set of pairs of formula occurrences in R and writing $\mathcal{L}$ for a set of jumps, we define

$$
\mathsf{seq}_\mathcal{L}(R) := \{\pi \in \mathsf{seq}(R) \mid \forall (F, F') \in \mathcal{L}, F >_\pi F'\}
$$

In the example above, the net with jump from $(A \,\mathfrak{P}\, B) \otimes C^\perp$ to C has only one sequentialization left: $\mathsf{seq}_{((A\mathfrak{P}B)\otimes C^\perp, C)}(R)$ corresponds to the left sequentialization of Example 2.2.
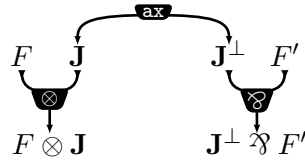
The aim of this paper is to provide a logical encoding of jumps in proof nets that validates the definition of $\mathsf{seq}_{\mathcal{L}}(R)$. More precisely, given a proof net R and a set of jumps $\mathcal{L}$, we want to define of proof net $\mathbf{J}_{\mathcal{L}}(R)$ such that $\mathsf{seq}(R) = \mathsf{seq}(\mathbf{J}_{\mathcal{L}}(R))$.

3 Jumps as axioms

In this section, we show how the graphical jumps presented above can be encoded in cut-free MLL proof structures using the axiom link of fresh atomic formulas $\mathbf{J}$ to create synchronisation points. We first introduce the idea for a single jump, then generalize it to multiple jumps.
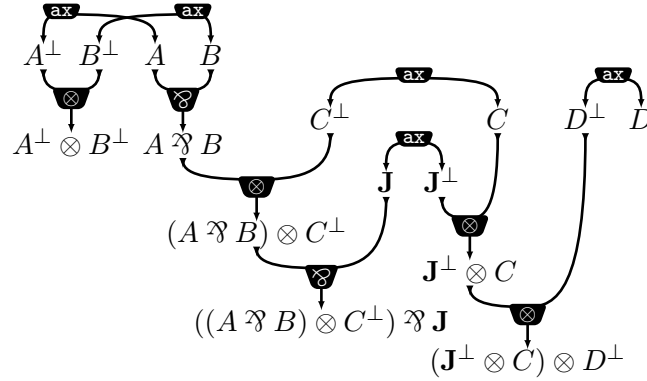
3.1 Case of a single jump

Definition 3.1 Let F and F' be two formulæ. A *jump* from F to F' is the gadget:

Let R be a proof-structure, and F, F' be two formula occurrences in R.

We define $\mathbf{J}_{F \to F'}(R)$, the *proof structure with a jump* from F to F', by grafting in R the jump from F to F' where F and F' are, and making all the types coherent by propagating downward the formula labels.

Let us illustrate the previous definition. Consider the (correct) proof-structure R in Example 2.2, we can add a jump from $F = (A \,\invamp\, B) \otimes C^{\perp}$ to C, resulting in the proof-structure $\mathbf{J}_{F \to C}(R)$, where the conclusion formulas have been updated to take into account the impact of adding the jump:

This proof-structure is much more rigid than R. It also has only two sequentializations:

$$
\cfrac{
  \cfrac{
    \cfrac{
      \cfrac{
        \cfrac{
          \cfrac{\vdash A^{\perp}, A \quad \vdash B^{\perp}, B}{\vdash A^{\perp} \otimes B^{\perp}, A, B} \otimes
        }{\vdash A^{\perp} \otimes B^{\perp}, A \,\invamp\, B} \invamp
        \quad \vdash C^{\perp}, C
      }{\vdash A^{\perp} \otimes B^{\perp}, (A \,\invamp\, B) \otimes C^{\perp}, C} \otimes
      \quad \vdash \mathbf{J}^{\perp}, \mathbf{J}
    }{\vdash A^{\perp} \otimes B^{\perp}, (A \,\invamp\, B) \otimes C^{\perp}, \mathbf{J}, \mathbf{J}^{\perp} \otimes C} \otimes
  }{\vdash A^{\perp} \otimes B^{\perp}, (A \,\invamp\, B) \otimes C^{\perp} \,\invamp\, \mathbf{J}, \mathbf{J}^{\perp} \otimes C} \invamp
  \quad \vdash D^{\perp}, D
}{\vdash A^{\perp} \otimes B^{\perp}, (A \,\invamp\, B) \otimes C^{\perp} \,\invamp\, \mathbf{J}, (\mathbf{J}^{\perp} \otimes C) \otimes D^{\perp}, D} \otimes
$$

and the one where the $\mathfrak{V}$ rule shown in green is commuted below the $\otimes$ rule. They differ in a much lighter way than the two sequentializations of R: indeed, forgetting every occurrence of $\mathbf{J}$ in both of these two sequent calculus proofs yields only one of the two sequentializations of the original proof-net. The presence of the jump from $F = (A \mathfrak{V} B) \otimes C^\perp$ to C forces F to be above C in every sequentialization of $\mathbf{J}_{F \to C}(R)$. Let us turn this example into a general theorem.

MLL$_\mathbf{J}$

From now on, we consider $\mathbf{J}$ as a special atom that does not appear in regular MLL formulae and proofs. We write MLL$_\mathbf{J}$ (respectively PS$_\mathbf{J}$) for the cut-free MLL proof system (resp. proof-structures) that does allow for the presence of $\mathbf{J}$ but with strong restrictions on its use described below.

We will call J-formulæ (respectively J-environments), denoted by $J, K, \dots$ (respectively $\Gamma_\mathbf{J}, \Delta_\mathbf{J}, \Theta_\mathbf{J}, \cdots$), formulæ in the grammar $J = \mathbf{J} \mid \mathbf{J}^\perp$ (respectively environments consisting of J-formulæ). MLL$_\mathbf{J}$ has the same rules as MLL except for J-formulae that are kept separated from other formulae until being used:

$$\frac{\vdash \Gamma, F, \Theta_\mathbf{J} \qquad \overline{\vdash \mathbf{J}, \mathbf{J}^\perp} \; \mathrm{ax_J}}{\vdash \Gamma, F \otimes \mathbf{J}, \mathbf{J}^\perp, \Theta_\mathbf{J}} \otimes_\mathbf{J}, F \notin J \qquad\qquad \frac{\vdash \Gamma, F, \mathbf{J}^\perp, \Theta_\mathbf{J}}{\vdash \Gamma, F \mathfrak{V} \mathbf{J}^\perp, \Theta_\mathbf{J}} \mathfrak{V}_{\mathbf{J}^\perp}, F \notin J$$

Proof structures and proof nets for MLL$_\mathbf{J}$ are essentially the same as in MLL except for the typing restriction put on links $\otimes$ and $\mathfrak{V}$ when dealing with J-formulæ. This motivates to define MLL$_\mathbf{J}$-formulas by the following grammar:

$$F_J = A \mid F_J \mathfrak{V} F_J \mid F_J \otimes F_J \mid F_J \mathfrak{V} \mathbf{J}^\perp \mid F_J \otimes \mathbf{J}.$$

The following proposition is trivial by induction on the structure of MLL$_\mathbf{J}$ proofs.

Proposition 3.2 *If $\vdash \Gamma$ is derivable in* MLL$_\mathbf{J}$*, then Γ can be partitioned in $\Delta, \Theta_\mathbf{J}$ such that every formula in Δ is a* MLL$_J$*-formula and every formula in $\Theta_\mathbf{J}$ is $\mathbf{J}^\perp$.*

In particular, when removing $\mathbf{J}$, MLL$_\mathbf{J}$ is equivalent to MLL. More precisely:

Definition 3.3 Let F be a MLL$_\mathbf{J}$-formula, the *erasure* $F|_\mathbf{J}$ of $\mathbf{J}$ in F is inductively defined by:

- if F is an atom: $F|_\mathbf{J} = F$;
- if $F = F_0 \square F_1$ with $\square \in \{\mathfrak{V}, \otimes\}$: $F_0|_\mathbf{J}$ if $F_1 \in \{\mathbf{J}, \mathbf{J}^\perp\}$, $F_0|_\mathbf{J} \square F_1|_\mathbf{J}$ otherwise;

Let π be an η-expanded sequent calculus proof in MLL$_\mathbf{J}$ (that is, all axiom rules are applied to atomic formulas only), the *erasure* $\mathsf{erase_J}(\pi)$ is inductively defined by:

- if π is an axiom $\vdash B^\perp, B$ then $\mathsf{erase_J}(\pi) = \pi$;
- if π is an application of a $\mathfrak{V}$ on $A \mathfrak{V} B$, with subproof π':
  · if $B \neq \mathbf{J}^\perp$, $\mathsf{erase_J}(\pi)$ is defined by applying a $\mathfrak{V}$ rule on $A|_\mathbf{J}$ and $B|_\mathbf{J}$ to $\mathsf{erase_J}(\pi')$;
  · if π is an application of a $\mathfrak{V}_{\mathbf{J}^\perp}$ after π' then $\mathsf{erase_J}(\pi)$ is defined as: $\mathsf{erase_J}(\pi')$;
- if π is an application of a $\otimes$ on $A \otimes B$, combining subproofs π_A and π_B:
  · if $B \neq \mathbf{J}$, then $\mathsf{erase_J}(\pi)$ is defined by combining $\mathsf{erase_J}(\pi_A)$ and $\mathsf{erase_J}(\pi_B)$ with a $\otimes$ rule on $A|_\mathbf{J}$ and $B|_\mathbf{J}$;
  · if π is an application of a $\otimes_\mathbf{J}$ combining π' and $\mathsf{ax_J}$ then $\mathsf{erase_J}(\pi)$ is defined as $\mathsf{erase_J}(\pi')$.

Let R be an MLL$_\mathbf{J}$-proof-structure the *erasure*, $\mathsf{erase_J}(R)$, is inductively defined in the same way as $\mathsf{erase_J}(\pi)$, that is by erasing every occurrence of $\mathbf{J}, \mathbf{J}^\perp$ and the $\mathfrak{V}$ and $\otimes$ link immediately connected to those atoms as well as the axioms links between $\mathbf{J}$ and $\mathbf{J}^\perp$.

Note that if $\vdash \Gamma$ is the conclusion of π (resp. R), then $\vdash \Gamma|_\mathbf{J}$ is the conclusion of $\mathsf{erase_J}(\pi)$ (resp. $\mathsf{erase_J}(R)$). Erasures cannot be empty by construction of MLL$_J$. Moreover, if $\pi \in \mathsf{seq}(R)$ then $\mathsf{erase_J}(\pi) \in \mathsf{seq}(\mathsf{erase_J}(R))$.

MLL is included in MLL$_\mathbf{J}$. Moreover:

Proposition 3.4 *If $\vdash \Gamma, \Theta_J$ is provable in $\mathsf{MLL_J}$ with $\vdash \Gamma \neq \emptyset$, then $\Gamma|_{\mathbf{J}}$ is provable in MLL. In fact, for $\pi \vdash \Gamma, \Theta_{\mathbf{J}}$, $\mathsf{erase_J}(\pi)$ is a MLL proof and there is an embedding $\mathsf{erase_J}(\pi) \hookrightarrow \pi$ of the rules of $\mathsf{erase_J}(\pi)$ onto the rules of π, preserving their kind and the order between them.*

Proof. By structural induction on π. Every case is straightforward, except maybe when $\otimes$ is the last rule of π: if F and G are the active formulae, one shall remark that by construction $F|_{\mathbf{J}}$ or $G|_{\mathbf{J}} \neq \emptyset$.   □

Theorem 3.5 *Let R be a proof-structure, and F and F' two formula occurrences, then $\mathsf{erase_J}(\mathsf{seq}(\mathbf{J}_{F \to F'}(R))) = \mathsf{seq}_{(F,F')}(R)$. In other words, the following diagram commutes:*

$$
\begin{array}{ccc}
\mathsf{PS_J} & \xrightarrow{\ \mathsf{seq}\ } & \mathcal{P}(\mathsf{MLL_J}) \\[4pt]
{\scriptstyle \mathbf{J}_{F \to F'}(\text{-})} \Big\uparrow & & \Big\downarrow {\scriptstyle \mathcal{P}(\mathsf{erase_J}(\text{-}))} \\[4pt]
\mathsf{PS} & \xrightarrow{\ \mathsf{seq}_{(F,F')}\ } & \mathcal{P}(\mathsf{MLL})
\end{array}
$$

In particular, correctness of the proof-net enriched with a jump is euivalent to existence of particular sequentializations:

Corollary 3.6 (correctness) *Let R be a proof-structure, and F and F' two formula occurrences, then $\mathbf{J}_{F \to F'}(R)$ is correct if and only if there exists a sequentialization of R such that F is above F'.*

Let us prove Theorem 3.5.

Proof. $\subseteq$. From Def. 3.3 it is immediate that $\mathsf{erase_J}(\mathsf{seq}(R)) \subseteq \mathsf{seq}(\mathsf{erase_J}(R))$. Now, let π be any sequentialization of $R_{F \to F'}$. π contains an axiom $\vdash \mathbf{J}^{\perp}, \mathbf{J}$, to which is applied a $\otimes$-rule tensoring J with F. So F has been created in the second branch leaving this $\otimes$-rule. At one point below, there is a $\mathfrak{N}$-rule between F' and $\mathbf{J}^{\perp}$. So the use of $F' \mathfrak{N} \mathbf{J}^{\perp}$ is below the creation of F in π, hence so is the use of F' in $\pi|_J$.

$\supseteq$. Let $\pi \in \mathsf{seq}_{(F,F')}(R)$. By hypothesis, the rule r introducing F is above the rule $\mathsf{r'}$ using F' in π, we can thus perform the following transformation adding three more rules in π (and propagating the changes in the resulting sequent accordingly):

$$
\pi =
\begin{array}{c}
\vdots \\
\hline \vdash \Gamma, F
\end{array}\!{\scriptstyle \mathsf{r}}
\quad
\begin{array}{c}
\vdots \\
\hline \vdash \Gamma', F'
\end{array}\!{\scriptstyle \mathsf{r'}}
\quad
\begin{array}{c}
\vdots
\end{array}
\qquad \rightarrow \qquad
\dfrac{\dfrac{\vdots}{\vdash \Gamma, F}{\scriptstyle \mathsf{r}} \quad \dfrac{}{\vdash \mathbf{J}, \mathbf{J}^{\perp}}{\scriptstyle \mathsf{ax}}}{\vdash F \otimes \mathbf{J}, \Gamma, \mathbf{J}^{\perp}}{\scriptstyle \otimes}
$$

$$
\begin{array}{c}
\vdots \\
\hline \vdash \Gamma'[F \otimes \mathbf{J}/F], F', \mathbf{J}^{\perp} \\
\hline \vdash \Gamma'[F \otimes \mathbf{J}/F], F' \,\mathfrak{N}\, \mathbf{J}^{\perp} \quad {\scriptstyle \mathfrak{N}} \\
\hline \quad {\scriptstyle \mathsf{r'}} \\
\vdots
\end{array}
$$

These 3 rules directly map to the jump gadget: the resulting proof is a sequentialization of $\mathbf{J}_{F \to F'}(R)$.□

We can now model a single jump in a proof net. This situation is far too restricted: we now move to the modelling of multiple jumps.

3.2   Multiple jumps

The definition of jump given above suffers from an obvious drawback: there is no canonical way to consider two jumps having the same source or destination. To solve this issue, we need the ability to

(i) parallelize: jumps with the same source or destination have no order, hence should not be represented as successive applications of $\mathfrak{N}$ or $\otimes$ rules;

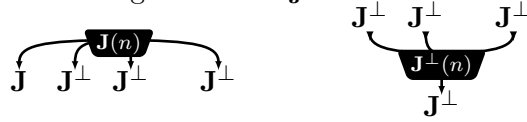(ii) contract: each jump needs two contractions: one on the source, one on the target.

Following these two requirements we ask **J** to have the following contraction properties:

$$\underbrace{\mathbf{J} \otimes \ldots \otimes \mathbf{J}}_{n \geq 0} \multimap \mathbf{J} \qquad \underbrace{\mathbf{J}^{\perp} \, \mathfrak{N} \ldots \mathfrak{N} \, \mathbf{J}^{\perp}}_{m > 0} \multimap \mathbf{J}^{\perp}$$

Hence, we disallow axioms on J-formulæ and extend $\mathsf{MLL_J}$ with the rules, for $n \geq 0, m > 0$:

$$\cfrac{}{\vdash \mathbf{J}, \underbrace{\mathbf{J}^{\perp}, \ldots, \mathbf{J}^{\perp}}_{m}} \; \mathbf{J}(m) \qquad\qquad \cfrac{\vdash \Gamma, \overbrace{\mathbf{J}^{\perp}, \ldots, \mathbf{J}^{\perp}}^{n}}{\vdash \Gamma, \mathbf{J}^{\perp}} \; \mathbf{J}(n)$$

and, in the same way, we add the following cells to $\mathsf{PS_J}$:

In the $\otimes_{\mathbf{J}}$ rule, we also replace $\mathrm{Ax_J}$ by $\mathbf{J}(n)$ for any $n \geq 0$.

Note that having a weakening on **J** (*i.e.* the rule $\mathbf{J}(0)$) is not required when dealing with cut-free proofs but we simply anticipate on the next section.

We generalise the encoding of a single jump to a list of jumps by making space to connect any edge with another one and then effectively realising the connections described in the list.

Definition 3.7 Given a proof-structure R, we define the *proof structure with synchronisation points* $\mathbf{J}(R)$ as the proof structure R in which every edge F of R is replaced by the following gadget $\mathbf{J}(F)$ (relabelling its successors to make all the types coherent):
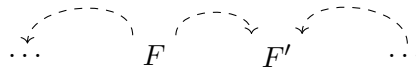

The $\otimes$-link bounded to the $\mathbf{J}(1)$ axiom is the *out-synchronisation point* of F, while the $\mathfrak{N}$-link bounded to the $\mathbf{J}^{\perp}(1)$ contraction is its *in-synchronisation point*.

Given two formula occurrences F and F' in R, a *jump from F to F'* is then encoded in $\mathbf{J}(R)$ by connecting the out-synchronisation point of F to the in-synchronisation point of F', that is, by adding a $\mathbf{J}^{\perp}$ edge from the $\mathbf{J}$ axiom link of $\mathbf{J}(F)$ to the $\mathbf{J}^{\perp}$ contraction link of $\mathbf{J}(F')$.

Finally, given $\mathcal{L}$ a set of pairs of formula occurrences in R, we define $\mathbf{J}_{\mathcal{L}}(R)$ the proof structure corresponding to $\mathbf{J}(R)$ with jumps between each pair of formula occurrences provided by $\mathcal{L}$.

Figure 4 illustrates the encoding of a jump from F to F' where F also has n outgoing jumps and F' also has m incoming jumps. In the sequel we will sometimes hide these jumps-as-axioms encoding and schematize them with dashed arrow, which in the case of Figure 4 would correspond to:

Remark 3.8 It is striking to note how close our encoding of jumps can be from double negation translations used to add sequentializations in the translation of classical logic into intuitionistic logic. Indeed, in linear logic, double negation translation correspond to $(A \multimap \bot) \multimap \bot$ that is $(A \otimes \mathbf{1}) \, \mathfrak{N} \perp$. This can be

understood as the production of a token (**1**) once A is created, and the wait for this token to return before A can be consumed. In the above encoding of jumps, the **J** atom replaced **1**, which allows many tokens to be produced and given to the environment after the creation of A (these tokens furthermore carry the information of coming from A thanks to the axiom link), while on the other side A can now wait for many specific tokens (coming from other formulae) to be put in the environment before it can actually be used.

The definition of $\mathsf{erase_J}\,(()\,\_)$ (see 3.3) does not change and following the same reasoning as for Theorem 3.5 we have:

Theorem 3.9 *Let R be a cut-free proof-structure, and $\mathcal{L}$ be a set of pairs of formula occurrences, then* $\mathsf{erase_J}\,(\mathsf{seq}\,(\mathbf{J}_{\mathcal{L}}(R))) = \mathsf{seq}_{\mathcal{L}}\,(R)$. *In other words,*

$$
\begin{array}{ccc}
\mathsf{PS_J} & \xrightarrow{\ \mathsf{seq}\ } & \mathcal{P}(\mathsf{MLL_J}) \\[4pt]
{\scriptstyle \mathbf{J}_{\mathcal{L}}(\_)}\Big\uparrow & & \Big\downarrow{\scriptstyle \mathcal{P}(\mathsf{erase_J}(\_))} \\[4pt]
\mathsf{PS} & \xrightarrow{\ \mathsf{seq}_{\mathcal{L}}\ } & \mathcal{P}(\mathsf{MLL})
\end{array}
$$

Corollary 3.10 (correctness) *Let R be a proof-structure, and $\mathcal{L}$ be a set of pairs of formula occurrences, then $\mathbf{J}_{\mathcal{L}}(R)$ is correct if and only if there exists a sequentialization of R such that F is above F' for every $(F, F') \in \mathcal{L}$.*

Remark 3.11 In [9], jumps are defined as edges from one link to one of the *entry port* of another link: a parr link having one entry port to which each of its premises are plugged; a tensor link having two entry ports, one for each of its premises. The correctness criterion for MLL proof structures with such graphical jumps is then a recast of the usual Danos-Regnier criterion: a proof structure is correct if and only if every of its *switching paths* is connected and acyclic, a switching path being here defined as a path which does not use any two edges entering trough the same port of a $\mathfrak{P}$ link.
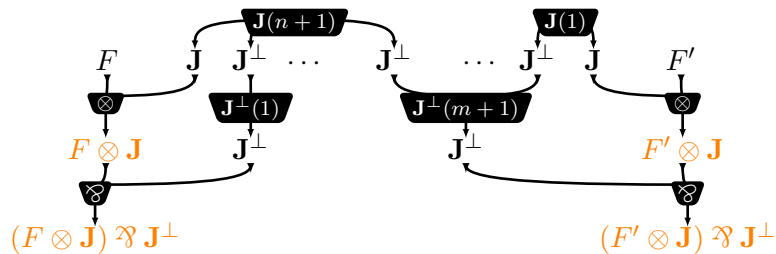
In the Danos-Regnier criterion for MLL proof structures, switching paths are defined as paths which never follow both premises of a $\mathfrak{P}$ link. Viewing our in-synchronisation point as their entry port, the usual Danos-Regnier criterion on our encoding of jumps correspond *exactly* to their recast of the same criterion.

Remark 3.12 In [15] Hughes introduces a MLL-encoding of his unification nets in order to prove their correction. Although we were not aware of it at the time we wrote this paper it must be acknowledged that this encoding is morally the same as our gadget from Definition 3.1 Being only interested in an encoding that preserves splitting tensors and switching cycles (*i.e.* correction), Hughes does not develop on the sequentialisation aspect of his encoding: in particular he shows no connection between his encoding and the set of sequentialisations of its nets, multiple cuts are not treated canonically and cuts are treated as tensors, evacuating any questions about their dynamics.

4 Cuts

We now investigate the internalisation of jumps in the presence of cuts. In particular, we need to define the dynamics of $\mathbf{J}$ and $\mathbf{J}^{\perp}$.
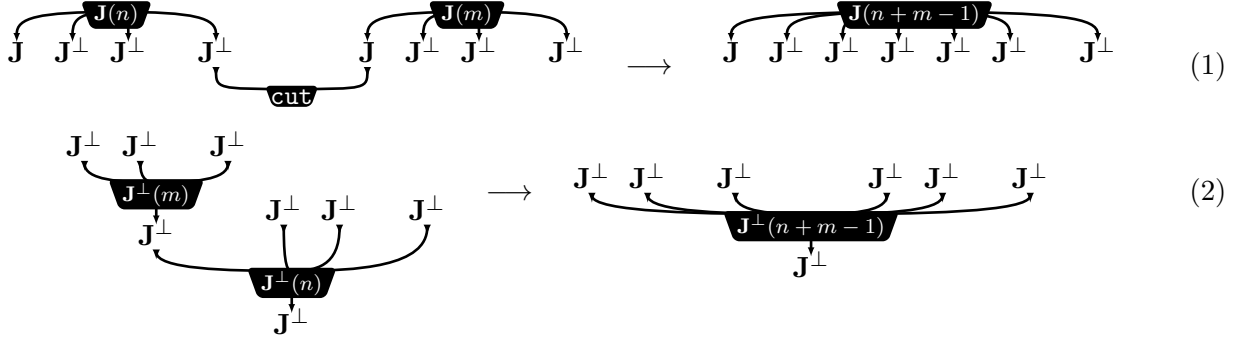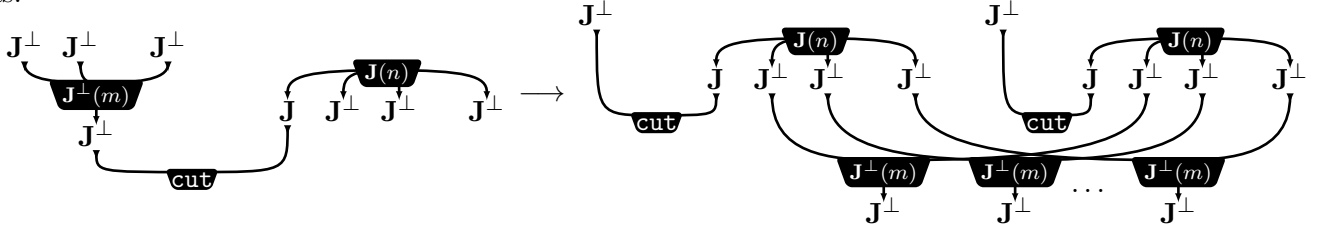

Fig. 4. Multiple jumps

Fig. 5. Reductions rules for propagating jumps

As intuited by [8], we expect jumps to propagate through cut elimination: suppose indeed that the left-hand side of a cut (on a formula F) has jumps coming into it, while the right-hand side ($F^\perp$) has jumps coming out of it. It forces, *in any sequentialization*, the sub-formulæ of the cut-formula to be above what is below the cut. This information has to be kept in some way during cut-elimination, even when the considered cut vanishes. Moreover, if n jumps enter into F and m jumps go out of $F^\perp$, the reduction of the cut will produce $m \times n$ jumps: one for each composition of jumps. This reveals the exponential nature of jumps: each jump arriving onto one of the two cut formulae must be propagated towards every formula occurrences reached by a jump leaving one of the two cut formulae, which cannot be realized by the multiplicative combinatorics. Unsurprisingly we define a commutation rule between $\mathbf{J}^\perp(m)$ and $\mathbf{J}(n)$ as:

which corresponds to the following map from the monoidal structure of any monoidal category interpreting MELL proof-structures (γ stands for the commutation):
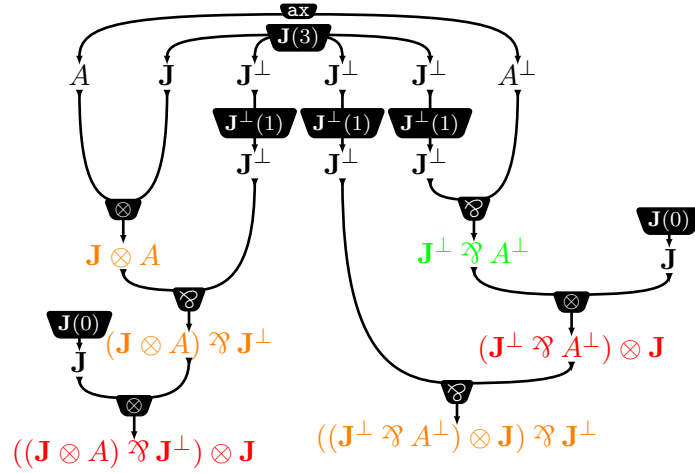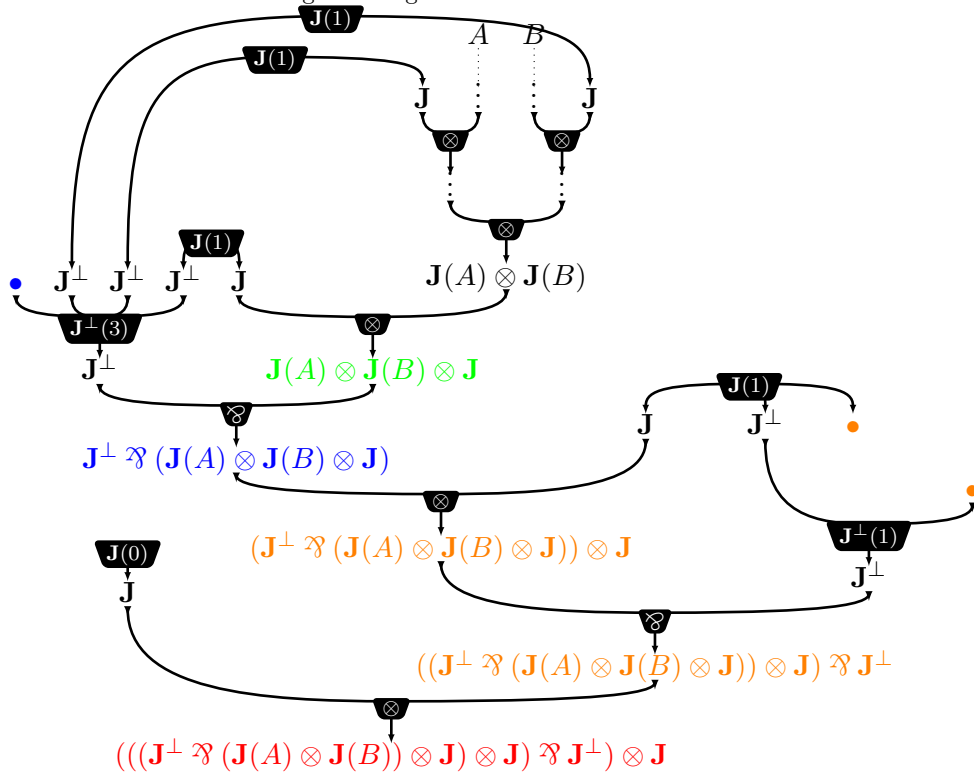
$$\mathsf{ctr}(m)^\perp \circ \gamma \circ \mathsf{ax}(n) = \bigotimes_n \mathbf{J} \multimap \bigotimes_n \bigotimes_m \mathbf{J} \multimap \bigotimes_m \bigotimes_n \mathbf{J} \multimap \bigotimes_m \mathbf{J}$$

To propagate newly created jumps from the above reduction rule, we also need $\mathbf{J}(n)$ and $\mathbf{J}^\perp(m)$ to both act as a monoid: the corresponding rules are given in Figure 5 ($m \geq 0, n > 0$).
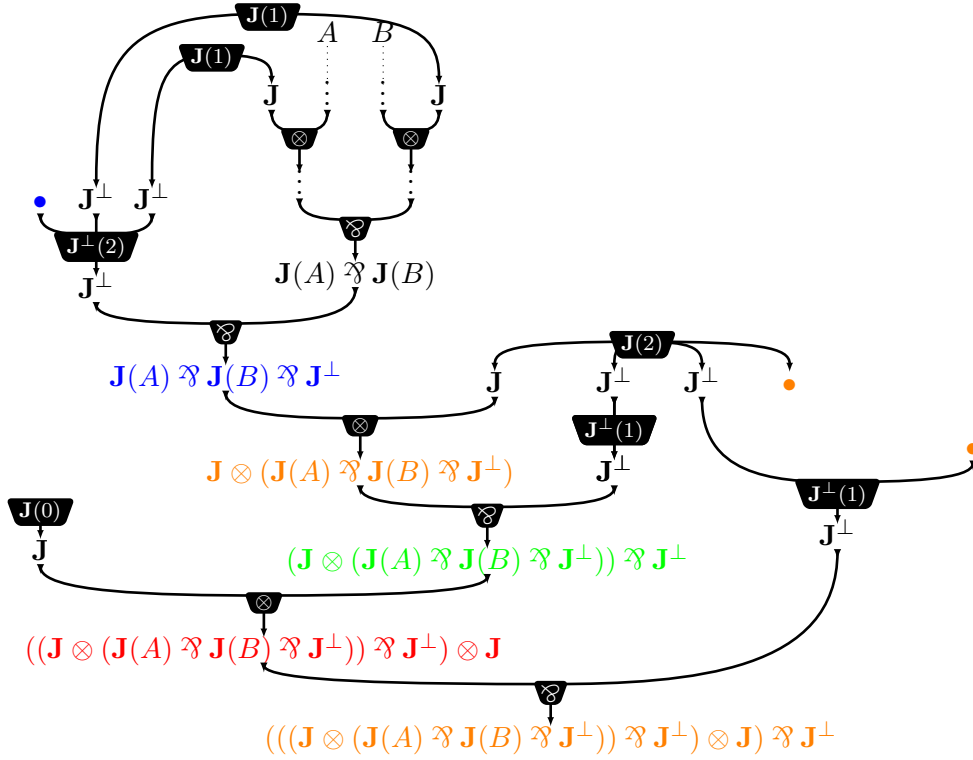
Bisimulation

On top of propagating the jumps arriving on the premises of two links involved in a cut, to the target of the jumps leaving the conclusions of these exact same links, we also need to propagate jumps with respect to the sub-formula ordering, and to destroy the ones that arrive onto the premises of a cut being eliminated.

With this intuition in mind, we extend the definition of $\mathbf{J}_\mathcal{L}(\_)$ presented in Section 3 in order to preserve the duality of (cut) formulae and ensure the propagation of jumps throughout cut elimination. We add propagation and absorption points to the previous synchronisation points:

Fig. 6. Gadget for the axiom A and $A^\perp$

Fig. 7. Gadget for a positive binary connective $A \otimes B$

Definition 4.1 Given a proof-structure R, we define the *proof structure with synchronisation points* $\mathbf{J}(R)$ as the proof structure R in which the outgoing edges of axiom, tensor ($\otimes$) and par ($\invamp$) links are replaced (from top to bottom) by the gadgets depicted in Figures 6, 7 and 8 respectively (making all the types coherent, $\mathbf{J}(A)$ is the type of gadget interpreting A).

- As before, orange links define the *out- and in-synchronisation points* of an edge F in R (for axiom links we have a unique out-synchronisation point placed on the positive edge, it is common to the two edges in order to avoid redundancy).

- Blue and Green links are *propagation* points, they will allow the incoming jumps of the premises of a link to inherit from the outgoing jumps of the link during cut elimination (or the outgoing jumps of the

Fig. 8. Gadget for a negative binary connective $A \,\mathregular{⅋}\, B$

conclusion of an axiom link to be transfered to the positive atom after reduction). In particular, blue links are connected to the out-synchronisation point of the premises of the link being replaced.

- Finally, Red links are *absorption* points, they will erase, the incoming jump of the premises of a cut after its elimination.

Given two formula occurrences F and F' in R, a *jump from F to F'* is now encoded in $J(R)$ by connecting the out-synchronisation point of F to the in-synchronisation point of F', *and*, if F' is the premise of a link in R, to the Blue propagation point of that link.

Let $\mathcal{L}$ be a set of pairs of formula occurrences in R, we note $\mathbf{J}_{\mathcal{L}}(R)$ the $\mathsf{MLL_J}$-proof structure corresponding to $\mathbf{J}(R)$ with all the jumps described in $\mathcal{L}$.

Remark 4.2 Although our encoding is quite heavy the resulting proof-structure has a number of links and edges which is linear in the number of links and edges of the original structure and the additional number of jumps: gadgets introduce at most 15 links and 22 edges for one link; and jumps introduced at most 2 new edges.

As intuited in the introduction of the section and detailed below, it is during cut elimination that our encoding may grow the most: in order not to lose any ordering our encoding propagates every jumps arriving onto the premises of a cut towards edges targeted by the jumps leaving these premises after cut elimination. In other words, eliminating a cut in R will amount to eliminate the corresponding gadgets in $\mathbf{J}(R)$ but may create $m \times n$ new edges where m is the number of jumps arriving onto the premises of the cut and n is the number of jumps leaving them. Hence, the number of edges in $\mathbf{J}(R)$ after cut elimination may be an exponential of the original number of jumps in the original number of cuts.

The propagation of all jumps is sufficient in order to get Theorem 4.3 but may generate unnecessary redundancy. The question of eliminating this redundancy is interesting in itself but is out of the scope of this paper, as this would probably involve changing the way cut elimination is performed and thus is left for future work.

On top of reintroducing duality between positive and negative formulæ with synchronisation points, absorption and propagation points also allow to destruct or transfer jumps during cut elimination. Indeed,

one can check that:

- dual formulæ are interpreted by dual formulæ;
- one step of cut reduction between F and $F^\perp$ in R, written $R \to_{F,F^\perp}$, can be simulated by several steps of cut reduction in $\mathbf{J}_\mathcal{L}(R)$: one has to eliminate all the cuts between their corresponding gadgets and jumps. We write $\mathbf{J}_\mathcal{L}(R) \to^*_{F,F^\perp}$ for the corresponding MLL$_\mathbf{J}$ proof structure.
- during these steps of cut elimination, jumps coming to one formula are merged with the jumps coming out of it and its dual, on both directions. Moreover premises of both formulæ also inherit the jump going out of these formulæ, or, in the case of axiom links, the inheritance is put on the positive atom.

Let us make precise how $R \to_{F,F^\perp}$ and $\mathbf{J}_\mathcal{L}(R) \to^*_{F,F^\perp}$ relate. For $\mathcal{L}_1, \mathcal{L}_2$ sets of jumps (we recall that jumps are pairs of formulæ), we define:

- (concatenation) $\mathcal{L}_1 \circ \mathcal{L}_2 = \{(F,G) \mid \exists H, (F,H) \in \mathcal{L}_1, (H,G) \in \mathcal{L}_2\}$
- (exponentiation) $\mathcal{L}_1 * \mathcal{L}_2 = \pi_1(\mathcal{L}_1) \times \pi_2(\mathcal{L}_2)$, where π_i are projection maps.

For $\mathcal{L}$ a set of jumps and F a formula occurrence, we define:

- (rules as jumps) $\mathcal{L}_F = \begin{cases} \{(F_1,F),(F_2,F)\} & \text{if } F = F_1 \square F_2 \\ \{(A,A^\perp)\} & \text{if } F \in \{A, A^\perp\} \text{ is an atom} \end{cases}$
- (in and out restrictions) $\mathcal{L}_{\to\mathcal{F}} = \{(G,F) \in \mathcal{L} \mid F \in \mathcal{F}\}\}$, $\mathcal{L}_{\mathcal{F}\to} = \{(F,G) \in \mathcal{L} \mid F \in \mathcal{F}\}$;
- (exclusion) $\mathcal{L}\backslash\mathcal{F} = L\backslash(\mathcal{L}_{\to\mathcal{F}} \cup \mathcal{L}_{\mathcal{F}\to})$;

In $\mathbf{J}_\mathcal{L}(R)$, the set $I_F = \mathcal{L}_F \cup (\mathcal{L} \circ \mathcal{L}_F)$ corresponds to the jumps toward the Blue propagation point of $\mathbf{J}(F)$. Cut elimination between F and $F^\perp$ thus creates the following jumps: $C_{F,F^\perp} = (I_F \cup I_{F^\perp}) * (\mathcal{L}_{F\to} \cup \mathcal{L}_{F^\perp\to})$ (as jumps in $\mathcal{L}_{F\to}$ are duplicated when their target is the premise of a link, the resulting jumps are duplicated in the same way) and delete the one in $\mathcal{L}_{\to F} \cup \mathcal{L}_{\to F^\perp}$ ($\mathbf{J}^\perp$ edges corresponding to $I_F \cup I_{F^\perp}$ are also deleted but they were only replications of remaining jumps in $\mathcal{L}$).

Writing $\mathcal{L}\downarrow\{F,F^\perp\}$ for $(\mathcal{L}\backslash\{F,F^\perp\}) \cup C_{F,F^\perp}$, we have:

Theorem 4.3 (bisimulation) *Let R be a proof-structure, $\mathcal{L}$ be a list of pairs of formula occurrences, and $F, F^\perp$ be the premises of a cut in R, then*

$$\mathbf{J}_\mathcal{L}(R) \to^*_{F;F^\perp} = \mathbf{J}_{\mathcal{L}\downarrow\{F,F^\perp\}}(R \to_{F;F^\perp})$$

Corollary 4.4 *Let R be a proof-structure and $\mathcal{L}$ be a list of pairs of formula occurrences, then*

$$\mathsf{seq}_{\Downarrow\mathcal{L}}(\Downarrow R) = \mathsf{seq}(\Downarrow (\mathbf{J}_\mathcal{L}R))$$

where $\Downarrow \mathcal{L}$ is defined as $\mathcal{L} \downarrow \rho(1) \cdots \downarrow \rho(n)$ for ρ any reduction sequence from R to $\Downarrow R$.

Proof. Theorem 4.3 implies $\mathbf{J}_{\Downarrow_\rho\mathcal{L}}(\Downarrow_\rho R) = \Downarrow_\rho (\mathbf{J}_\mathcal{L}R)$ for any reduction sequence ρ, and by confluence of cut elimination in proof-structure the reduction path does not matter.
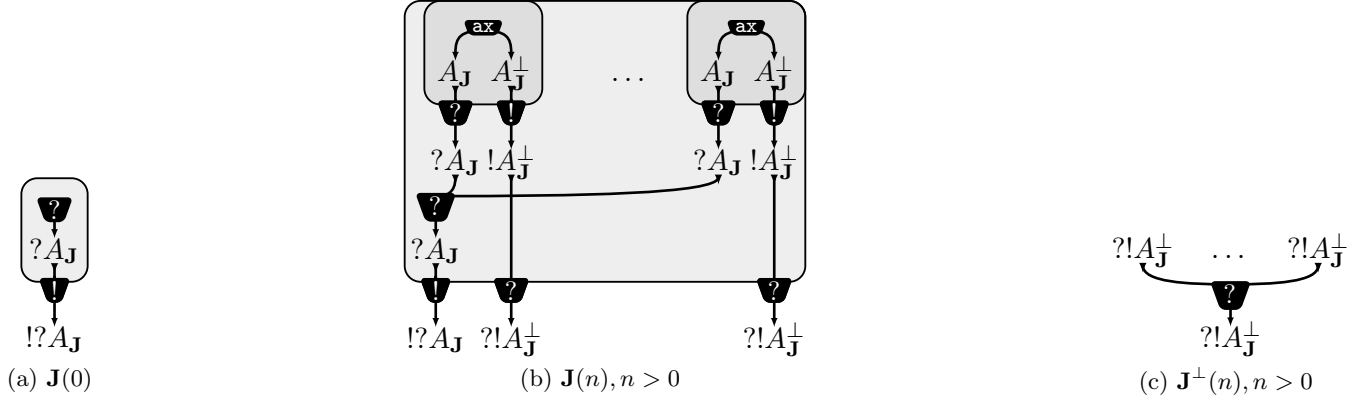
Back in a cut-free context, one can apply Theorem 3.9 to get the result. □

Correction

As done for tensor in Section 3, we restrict the use of cuts on J-formulae in order to have a well-defined notion of erasure of $\mathbf{J}$ from MLL$_\mathbf{J}$ proofs to MLL proofs:

$$\cfrac{\cfrac{}{\vdash \mathbf{J}, \mathbf{J}^\perp, \cdots, \mathbf{J}^\perp} \text{ ax}_\mathbf{J}(n) \qquad \vdash \Gamma, \mathbf{J}^\perp, \Theta_\mathbf{J}}{\vdash \Gamma, \mathbf{J}^\perp, \cdots, \mathbf{J}^\perp, \Theta_\mathbf{J}} \text{ cut}_\mathbf{J}$$

MLL$_\mathbf{J}$ is closed under cut elimination and following the same reasoning as before we still have

Fig. 9. Encoding of the combinators on $\mathbf{J}$ and $\mathbf{J}^{\perp}$

Theorem 4.5 *Let R be a proof-structure, and $\mathcal{L}$ be a set of pairs of formula occurrences, then*

$$\mathsf{erase}_{\mathbf{J}}\left(\mathsf{seq}\ \left(\mathbf{J}_{\mathcal{L}}(R)\right)\right) = \mathsf{seq}_{\mathcal{L}}\ (R).$$

So as before we have an encoding $(\mathbf{J}_{\mathcal{L}})$ of graphical jumps in $\mathsf{MLL}_{\mathbf{J}}$ (and its corresponding decoding $\mathsf{erase}_{\mathbf{J}}$) such that:

$$
\begin{array}{ccc}
\mathsf{PS}_{\mathbf{J}} & \xrightarrow{\ \mathsf{seq}\ } & \mathcal{P}(\mathsf{MLL}_{\mathbf{J}}) \\
\mathbf{J}_{\mathcal{L}}(\text{-})\Big\uparrow & & \Big\downarrow \mathcal{P}(\mathsf{erase}_{\mathbf{J}}(\text{-})) \\
\mathsf{PS} & \xrightarrow{\ \mathsf{seq}_{\mathcal{L}}\ } & \mathcal{P}(\mathsf{MLL})
\end{array}
$$

and, furthermore, by Theorem 4.3, there exists a bisimulation procedure so that the above diagram is stable under cut-elimination.

5   Encoding in MELL + Mix

In this section, in order to finish our logical internalization of jumps, we eventually show how jumps, as defined in Figure 3 and 4, can be implemented in the exponential fragment of $\mathsf{MELL} + \mathsf{Mix}$.

As explained in Section 3, we need $\mathbf{J}$ and $\mathbf{J}^{\perp}$ to be able to contract. This can be achieved using the contraction rule of the ?-exponential in MELL, so our encoding of $\mathbf{J}$ and $\mathbf{J}^{\perp}$ must involve this exponential modality. $\mathbf{J}$ only needs the ability to contract at the level of axioms while $\mathbf{J}^{\perp}$ is free to contract everywhere, furthermore, $\mathbf{J}$ and $\mathbf{J}^{\perp}$ must be dual from each other, as such, we set:

$$\mathbf{J} = {!?}A_{\mathbf{J}} \qquad\qquad \mathbf{J}^{\perp} = {?!}A_{\mathbf{J}}^{\perp}$$

where $A_{\mathbf{J}}$ is a special atom that does not appear in regular MLL formulae.

In Figure 9 we show how the static properties of jumps (that are $\mathbf{J}(n \geq 0)$ and $\mathbf{J}^{\perp}(m > 0)$) can be derived via this encoding. The dynamics properties of jumps given described in Section 4 can then be derived from the usual dynamics of exponential in MELL.

6   Conclusion and future work

We have presented a logical framework, $\mathsf{MLL}_{\mathbf{J}}$, in which MLL-proof-structures with jumps can be encoded. This framework can be seen both as an extension of MLL with a special atom and as a fragment of MELL + Mix. As such, correctness criteria for proof-structures in these logics induce correctness criteria for $\mathsf{MLL}_{\mathbf{J}}$-proof-structures. In particular, the Danos-Regnier correctness criterion for MLL induces correctness

criterion for $\mathsf{MLL_J}$, which corresponds to the correctness criterion given in [8], in a setting where jumps in proof structures are treated from a purely graph-theoretical point of view.

Our encoding is compatible with the dynamics of proof-nets: cut elimination in our framework achieves the graphical dynamics of jumps considered in [8,7]. On top of being the most natural one (it makes two jumps to compose if one targets the premise of a link involved in a cut and the other leaves one of these links), we believe that this dynamics is justified by the following conjecture:

Conjecture 6.1 *Let R be a MLL proof-net, and $\mathcal{L}$ be a set of pairs of formula occurrences such that* $\sec_{\mathcal{L}}(R) = \{\pi\}$, *i.e. $\mathcal{L}$ adds enough constraints on R for its sequentialization set to be a singleton, then*

$$\sec(\Downarrow \mathbf{J}_{\mathcal{L}}(R)) = \{\pi' \mid \exists \rho, \pi' =\Downarrow_{\rho} \pi\}$$

In other words, the freedom of sequentialization induced by the cut elimination procedure in Jumped proof structures corresponds exactly to the choices made during cut elimination in the sequent calculus. $\supseteq$ is a consequence of Theorem 4.3, $\subseteq$ however is left as future work.

Our framework draws a clear line between formulæ encoding jumps and regular MLL formulæ. This advocates for easy extensions to other logics such as MALL or MELL with or without the Mix rule. In particular we plan, in expanding this work to MELL to study in detail the correspondence between the dynamics that is induced by our modelling of jumps and the cut-elimination considered in [7] for the exponentials as well as how the notion of cones introduced by di Giamberardino as a replacement and generalization of exponential boxes translates in our setting.

This work fits in a general tendency to try to make sense logically of the graph-theoretic foundations of proof-nets. Here, we reinterpreted jumps as a purely logical structure.

References

[1] Accattoli, B. and S. Guerrini, *Jumping boxes*, in: *CSL*, volume 5771 of *Lecture Notes in Computer Science*, pages 55–70, Springer (2009).
https://doi.org/10.1007/978-3-642-04027-6_7

[2] Baelde, D., A. Doumane and A. Saurin, *Infinitary proof theory: the multiplicative additive case*, in: *CSL*, volume 62 of *LIPIcs*, pages 42:1–42:17, Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2016).
https://doi.org/10.4230/LIPIcs.CSL.2016.42

[3] Buss, S. R., *An introduction to proof theory* **137**, pages 1–78 (1998).

[4] Chouquet, J. and L. V. Auclair, *An application of parallel cut elimination in multiplicative linear logic to the Taylor expansion of proof nets*, Logical Methods in Computer Science **Volume 17, Issue 4** (2021).
https://doi.org/10.46298/lmcs-17(4:22)2021

[5] Curien, P. and C. Faggian, *L-nets, strategies and proof-nets*, in: C. L. Ong, editor, *Computer Science Logic, 19th International Workshop, CSL 2005, 14th Annual Conference of the EACSL, Oxford, UK, August 22-25, 2005, Proceedings*, volume 3634 of *Lecture Notes in Computer Science*, pages 167–183, Springer (2005).
https://doi.org/10.1007/11538363_13

[6] Curien, P. and C. Faggian, *An approach to innocent strategies as graphs*, Inf. Comput. **214**, pages 119–155 (2012).
https://doi.org/10.1016/j.ic.2011.12.006

[7] Di Giamberardino, P., *Jump from parallel to sequential proofs: exponentials*, Mathematical Structures in Computer Science **28**, page 1204–1252 (2018).
https://doi.org/10.1017/S0960129516000414

[8] Di Giamberardino, P. and C. Faggian, *Jump from parallel to sequential proofs: Multiplicatives*, in: Z. Ésik, editor, *Computer Science Logic*, pages 319–333, Springer Berlin Heidelberg, Berlin, Heidelberg (2006), ISBN 978-3-540-45459-5.
https://doi.org/10.1007/11874683_21

[9] Di Giamberardino, P. and C. Faggian, *Proof nets sequentialisation in multiplicative linear logic*, Annals of Pure and Applied Logic **155**, pages 173–182 (2008), ISSN 0168-0072.
https://doi.org/10.1016/j.apal.2008.04.002

[10] Faggian, C. and F. Maurel, *Ludics nets, a game model of concurrent interaction*, in: *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*, pages 376–385, IEEE Computer Society (2005).
https://doi.org/10.1109/LICS.2005.25

[11] Girard, J., *Linear logic*, Theor. Comput. Sci. **50**, pages 1–102 (1987).
https://doi.org/10.1016/0304-3975(87)90045-4

[12] Girard, J., *Locus solum: From the rules of logic to the logic of rules*, Math. Struct. Comput. Sci. **11**, pages 301–506 (2001).
https://doi.org/10.1017/S096012950100336X

[13] Girard, J.-Y., *Quantifiers in linear logic II*, in: *Nuovi problemi della logica e della filosofia delle scienze*, volume II of *CLUEB*, pages 79–89 (1991).

[14] Girard, J.-Y., *Proof-nets: The parallel syntax for proof-theory*, in: *Logic and Algebra*, pages 97–124, Marcel Dekker (1996).
https://doi.org/10.1201/9780203748671-5

[15] Hughes, D. J. D., *Unification nets: Canonical proof net quantifiers*, in: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, page 540–549, Association for Computing Machinery, New York, NY, USA (2018), ISBN 9781450355834.
https://doi.org/10.1145/3209108.3209159

[16] Hughes, D. J. D. and R. J. van Glabbeek, *Proof nets for unit-free multiplicative-additive linear logic*, ACM Trans. Comput. Log. **6**, pages 784–842 (2005).
https://doi.org/10.1145/1094622.1094629

[17] Laurent, O., *Polarized proof-nets: Proof-nets for LC*, in: *TLCA*, volume 1581 of *Lecture Notes in Computer Science*, pages 213–227, Springer (1999).
https://doi.org/10.1007/3-540-48959-2_16