Correctness Criteria for Second Order Multiplicative Linear Logic

Adrien Ragot, Thomas Seiller, Lorenzo Tortora de Falco

Correctness Criteria for Second Order Multiplicative Linear Logic

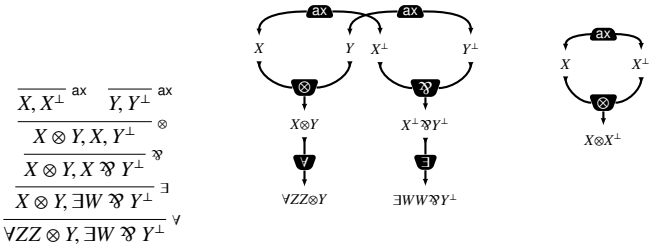
Adrien Ragot
Université Sorbonne Paris Nord - LIPN
Universitá Degli Studi Roma Tre
Villetaneuse, France
ragot@lipn.fr

Thomas Seiller
Université Sorbonne Paris Nord
UMR 7030, LIPN, CNRS
Villetaneuse, France
seiller@lipn.fr

Lorenzo Tortora de Falco
Universitá Degli Studi Roma Tre
Istituto Nazionale di Alta Matematica, GNSAGA
Roma, Italy
tortora@uniroma3.it

***Abstract*—We define untyped proof structures for second order multiplicative linear logic as hypergraphs that we call *nets*. We give a desequentialisation method associating a proof-tree its representing net, and because not all nets represent a proof we define a correctness criterion, based on parsing, which determines (and construct) the proof that is represented by a net S (if such proof exists). Because the parsing rewriting is confluent on proof nets and always decreases the size of a net our criterion runs in quadratic time.**

***Index Terms*—Linear Logic, Proof nets, Second order quantification**



$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{X, X^\perp}\,\mathsf{ax} \quad \dfrac{}{Y, Y^\perp}\,\mathsf{ax}}{X \otimes Y, X, Y^\perp}\,\otimes}{X \otimes Y, X \parr Y^\perp}\,\parr}{X \otimes Y, \exists W\, W \parr Y^\perp}\,\exists}{\forall Z Z \otimes Y, \exists W\, W \parr Y^\perp}\,\forall$$

Fig. 1: A sequent calculus proof (left) its corresponding proof net (center). On the right: a graph not representing any proof.

I. Introduction

Linear logic, introduced by Jean-Yves Girard [1], refines both intuitionistic and classical logic. It emerged from a meticulous study of a denotational model of lambda calculus [2], where the intuitionistic implication was decomposed into two distinct operations. From its inception, linear logic has been characterized by two complementary representations: a sequent calculus proof system and a graphical formalism called *proof nets*.

The conceptual foundation of proof nets parallels natural deduction, identifying proofs up to rule commutations and offering a more intuitive framework for understanding proofs as programs. Unlike natural deduction, however, the syntax of proof nets is inherently permissive. Identifying the subset of graphical representations that correspond to valid sequent calculus proofs requires specific *correctness criteria*. These criteria have been extensively studied since Girard's original work (such as [3], [4], [5]), and they typically come with a *sequentialization theorem*—a result that reconstructs a sequent calculus proof from a graphical representation satisfying the correctness criteria. For instance, Figure 1 illustrates two graphical proofs and a sequent calculus proof. The central graphical proof qualifies as a proof net, representing the sequent calculus proof on the left-hand side, while the right hand-side graphical proof fails the correctness criteria and does not correspond to any sequent calculus proof.

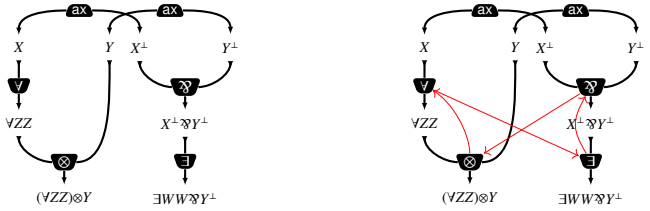
In the simplest setting of multiplicative linear logic (the smallest fragment), correctness criteria are entirely local [6], [7], [8]. A graphical proof is correct if it satisfies properties determined solely by the relationships between vertices and their immediate neighbors in the graph. Even in this restricted case, various criteria have been proposed, addressing in particular the computational complexity of verifying correctness. Extensions to larger fragments, incorporating additive connectives or exponentials, have inspired many new approaches (such as parsing criteria of [9], [3]), motivated both by complexity considerations and the desire to avoid global correctness criteria.

One noticeable gap in this body of work is the treatment of second-order quantifiers. While Girard studied correctness criteria for first-order quantifiers [10], these approaches are limited in scope, as they apply only to closed formulas and impose significant computational costs. This paper addresses this gap by proposing a definition of proof nets for second-order quantifiers, restricted to the simplest non-trivial fragment: second-order multiplicative linear logic. We introduce a definition of proof nets for second-order quantifiers, accompanied by a correctness criterion that is quadratic in the number of edges in the graphical representation. We prove a sequentialization theorem for this criterion and refine it further to develop a second, static correctness criterion.

Our approach employs a notion of graphical proofs that eschews type annotations, advancing beyond traditional methods. This "Curry-style" approach contrasts with the "Church-style" reliance on types, unifying proofs such as the identity at all types under a single representation. For instance, the lambda-term $\lambda x.x$ embodies the identity across all types. While omitting types introduces challenges in intuitive interpretation—allowing for generalized axiom rules that incorporate arbitrary sequences of formulas— it compels explicit handling of dependencies otherwise implicit in types.

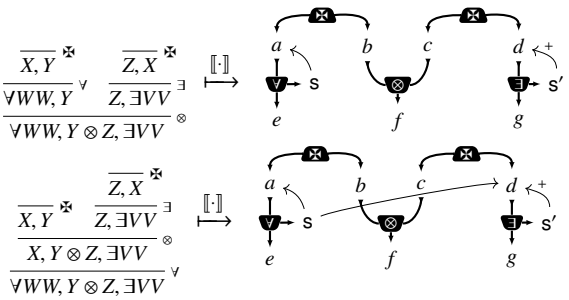
Consider the graphical proofs in Figure 2, when types are removed, dependencies—such as those introduced by the universal quantifier—become obscured, and two graphical proofs with a different correctness status are identified to the same object.

Even within the narrow scope of second-order multiplicative logic and allowing types, the interplay between quantifiers and multiplicative connectives presents challenging questions. Take for instance the following graphical proof:



Relatively to the multiplicative correctness criterion of Danos-Regnier the graphical proof above on the left side (denoted $S_{\overline{\eta}}$) is correct, however the quantifiers are not introduced properly. Indeed, the ⅋-rule cannot be applied before the tensor rule but in order to apply the ⊗-rule one must first apply the ∀-rule. However the ∀-rule cannot be applied first because its variable occurs free at the right side of the axiom, thus requiring the ∃-rule to be performed beforehand, but this rule can only be applied after the ⅋-rule. This cycle of dependencies, represented by red arrows in the figure on the right side, shows that the graphical proof on the left side, although correct for Danos-Regnier switchings, does not correspond to a proof.

To address this, we reintroduce dependencies using pointers, analogous to "jumps" in Girard's criteria for quantifiers [10]. However, our approach is more parsimonious: we introduce ∀-pointers to leaves (conclusions of axioms) and ∃-pointers to occurrences of existential witnesses. Avoiding types will allow us to distinguish proofs based on the use of their variables. For example the sequent calculus proof below can be represented by nets where the universal pointer goes above the exists only when representing the second proof tree (this will force the tensor to be sequentialized before the ∀-link), whereas in the case of the first proof tree that pointer does not appear: this is because the variable of the ∀-quantifier can be renamed before applying the tensor rule hence the two occurrences of X are not be related in the first proof tree.



Another line of work exists proposed by L. Strassburger and related deep inference [11]. Our work differs significantly from that of L. Strassburger because its MLL_2 proof structures, are inherently typed and use only atomic axioms [11]. Unlike Strassburger, we aim to study correctness independently of types, focusing instead on properties such as the geometry of the proof structures (that we will call net in this document). For example, Strassburger's use of implicit boxes complicates capturing proof equivalence, whereas our approach, similar to Girard's use of pointers [10], avoids boxes and resolves this issue. Furthermore, our parsing criterion is both novel and efficient, running in quadratic time, unlike the exponential-time complexity of existing criteria based on switchings such as [10] and [11].

II. Preliminaries

In this section we introduce hypergraphs equipped with pointers, in order to handle second order quantification and to interpret proofs of second order multiplicative linear logic. The challenge in the definition is to add quantifiers while keeping an untyped setting. Pointers are used to reintroduce exactly the needed information to deal with quantifiers (which is usually implicitly given by types).

Before going into these technical details involving quantifiers, we recall some basic definitions and results on proof nets for multiplicative linear logic. This will also allow us to introduce notations used in the remaining parts of the paper.

A. Second-order multiplicative linear logic

We quickly recall the syntax of second-order multiplicative linear logic.

Formulas of MLL_2 are defined inductively, given a countable set of propositional variables Var coming with an involution $(\cdot)^\perp : X \mapsto X^\perp$:

$$A, B \doteq X, X^\perp \mid A \otimes B \mid A \parr B \mid \forall X A \mid \exists X A$$

A *(one-sided) sequent* Γ is a finite sequence of formulas of MLL_2. The rules of the sequent calculus of MLL_2 are given in Figure 3. A *proof* of MLL_2 is a tree constructed using the rules of the sequent calculus. The rule introducing the existential quantifier as for premisse a sequent of the form $\Gamma, A[X \leftarrow B]$, as usual, the substitution $A[X \leftarrow B]$ must be stable under α–conversion thus B cannot contain variable which occurs bounded in A. An occurence of a formula A is *principal* in a rule of MLL_2 if it is introduced by an axiom rule (or a daimon rule subsection II-C) or is the main formula of the rule (these are the red occurences in Figure 3, for example $A \otimes B$ is the main formula of the tensor rule). Occurences of a same formula in a proof are related by the *trace function* tr, which allows to define a *thread* i.e. a sequence of occurences $x, \mathsf{tr}(x), \mathsf{tr}^2(x), \ldots$ An (occurence of a) formula y is *available* in a proof π if there exists a thread $x, \ldots, y$ such that x is principal in a rule of π.

B. Multiplicative nets

As in the work [12] we present *nets* as hypergraphs. We recall the basic definitions.

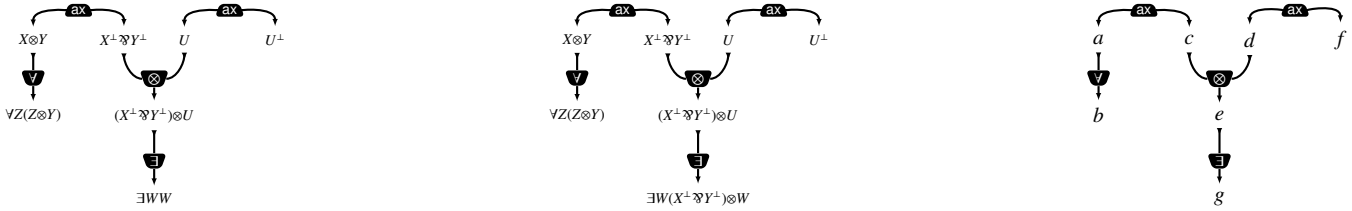


Fig. 2: Two typed nets that become the same net when forgetting types. The first net contains a forall-rule which is valid because X does not occur free in the conclusions, on the other hand the forall-rule cannot be valid in the second net because its variable X appears free in the conclusions. To distinguish the untyped nets which represent a proof and those that don't we must therefore consider additional information.

$$\frac{}{A, A^\perp}\ \text{ax} \qquad \frac{\Gamma, A, B, \Delta}{\Gamma, B, A, \Delta}\ \text{ex} \qquad \frac{\Gamma, A, B}{\Gamma, A \parr B}\ \parr \qquad \frac{\Gamma, A \quad \Delta, B}{\Gamma, A \otimes B, \Delta}\ \otimes$$

$$\frac{\Gamma, A \quad \Delta, A^\perp}{\Gamma, \Delta}\ \text{cut} \qquad \frac{\Gamma, A[Y/X]}{\Gamma, \forall XA}\ \forall^\star \qquad \frac{\Gamma, A[B/X]}{\Gamma, \exists XA}\ \exists$$

Fig. 3: Rules of the sequent calculus MLL_2.
($\star$) To apply the rule $\forall$, the variable Y must not appear free in Γ.

1) Nets and modules: Given a set X we will let $X^{\circledast}$ denote the set of finite sequences of elements of X without repetitions. Such sequences will be denoted by vectors $\vec{x} = x_1 x_2 \dots x_k$. In the following, we may refer to $\vec{x}$ as a *set* to denote the underlying set $|\vec{x}| = \{x_1, x_2, \dots, x_k\}$.

We fix a countable set of *positions* that we denote $\mathcal{P}$ which will define the nodes of our hypergraphs.

Definition 1. Suppose given a set L of *labels*. A *directed (L-labelled) hypergraph* is a tuple $(V, E, \mathrm{s}, \mathrm{t}, \ell)$ where $V \subset \mathcal{P}$ is a finite set of *positions* and E is a finite set of *links*, $\mathrm{s} : E \to V^{\circledast}$ is the *source map*, $\mathrm{t} : E \to V^{\circledast}$ is the *target map* and $\ell : E \to L$ is the *labelling* map.

A *link* is an hypergraph (V, E, s, t, ℓ) such that $E = \{e\}$ and $V = |s(e)| \cup |t(e)|$.

An hypergraph (V, E, s, t, ℓ) is *trivial* when $E = \emptyset$.

Notation 2. A *link* will be denoted as $\langle \vec{u} \rhd_l \vec{v} \rangle$, where $E = \{e\}$, $V = |\vec{u}| \cup |\vec{v}|$, $\mathrm{s}(e) = \vec{u}$, $\mathrm{t}(e) = \vec{v}$ and $\ell(e) = \ell$. A trivial hypergraph $(\{p_1, \dots, p_n\}, \emptyset, s, t, \ell)$ will be denoted by $\langle p_1, \dots, p_n \rangle$.

As a convention, we assume all the hypergraphs to be loop–free i.e. the source and target sets of each link are disjoint. In particular, $\vec{u}$ and $\vec{v}$ will be assumed disjoint when writing $\langle \vec{u} \rhd_l \vec{v} \rangle$. A *conclusion/output* (resp. a *premise/input*) of a directed hypergraph $\mathcal{H}$ is a position which is not in the source set (resp. target set) of any link in $\mathcal{H}$.

Notation 3. Given two sets X_0 and X_1 we denote their disjoint union $X_0 \uplus X_1$. Given two functions $f : X_0 \to E$ and $g : X_1 \to E$, we denote by $f \uplus g$ their *co-pairing*, i.e. the function taking an element $x \in X_0 \uplus X_1$, and returning either $f(x)$ if $x \in X_0$, or $g(x)$ if $x \in X_1$.

Definition 4. Given two hypergraphs $\mathcal{H}_1 = (V_1, E_1, \mathrm{t}_1, \mathrm{s}_1, \ell_1)$ and $\mathcal{H}_2 = (V_2, E_2, \mathrm{t}_2, \mathrm{s}_2, \ell_2)$, we define their sum as:

$$\mathcal{H}_1 + \mathcal{H}_2 = (V_1 \cup V_2, E_1 \uplus E_2, \mathrm{t}_1 \uplus \mathrm{t}_2, \mathrm{s}_1 \uplus \mathrm{s}_2, \ell_1 \uplus \ell_2).$$

Remark 5. Vertices may overlap in a sum (as we take the union of vertex sets rather than the disjoint union). As a consequence, a position may be input (or output) of several distinct links. Any hypergraph can therefore be defined as a sum of links and trivial hypergraphs, although not in a unique manner. We will define *the* representation of a hypergraph $\mathcal{H} = (V, E, s, t, \ell)$ as the sum

$$\sum_{p \in \mathrm{Is}(\mathcal{H})} \langle p \rangle + \sum_{e \in E} \langle s(e) \rhd_{\ell(e)} t(e) \rangle,$$

where $\mathrm{Is}(\mathcal{H})$ is the set of *isolated positions* in $\mathcal{H}$, i.e. elements $v \in V$ which are neither source or target of any link.

A *module* is an hypergraph such that for each position p there exists *at most* one link e such that $p \in |\mathrm{s}(e)|$, and *at most* one link e' such that $p \in |\mathrm{t}(e')|$. A *net* is an hypergraph such that for each position p there exists *at most* one link e such that $p \in |\mathrm{s}(e)|$, and *exactly* one link e' such that $p \in |\mathrm{t}(e')|$.

2) Nets for MLL: We now fix the set of labels as the set consisting of the *axiom* (ax), the tensor ($\otimes$), the parr ($\parr$), and the cut (cut) symbols. Furthermore we fix a family of links associated with those labels, namely:

- ax-labelled links that have no inputs (they are initial links) and exactly two outputs,
- cut-labelled links that have exactly two inputs and no outputs (they are final links),
- $\otimes$- and $\parr$-labelled links that have exactly two inputs and one output.

Formally we fix a countable set $\mathcal{P}$ of positions and a family of links $\mathcal{L}_{\mathsf{MLL}}$ defined as:

$$\mathcal{L}_{\mathsf{MLL}} = \left\{ \begin{array}{l} \langle p_1, p_2 \rhd_\otimes p_3 \rangle, \langle \rhd_{\mathsf{ax}} p_1, p_2 \rangle, \\ \langle p_1, p_2 \rhd_\parr p_3 \rangle, \langle p_1, p_2 \rhd_{\mathsf{cut}} \rangle \end{array} \middle| \, p_1, p_2, p_3 \in \mathcal{P} \right\}$$

Definition 6. A MLL *net* is a sum of links in $\mathcal{L}_{\mathsf{MLL}}$ which is a net.

Since the inception of LL it is well known that proofs from the sequent calculus may be represented as nets (or *proof structures*) [13]. One can easily define the net $[\![\pi]\!]$ associated with a sequent calculus proof π inductively. We

do not recall this definition here, as it is a particular case of the interpretation of second-order proofs found in section III.

However not all nets are the image of a proof. Among nets, one therefore distinguish *proof nets*, which are those nets $\mathcal{H}$ for which there exists a sequent calculus proof π with $\mathcal{H} = [\![\pi]\!]$.

The *correctness criteria* are properties of a net $\mathcal{H}$ which are necessary and sufficient for $\mathcal{H}$ to be a proof net. Those results are usually established through a *sequentialisation theorem* reconstructing from the considered property a sequent calculus proof π such that $\mathcal{H} = [\![\pi]\!]$.

We briefly discuss some of the known criteria for multiplicative linear logic (MLL). We first consider the most famous one, introduced by Danos and Regnier [7] as a simplification of Girard's original criterion. This criterion consists in taking a net S, compute its *switching graphs* σS, and check if all of these graphs are acyclic and connected. A switching graph is defined by disconnecting each $\parr$–link from one of its inputs.

Theorem 7 ([7]). *Given a* MLL *net S: Each switching σS of S is an acyclic and connected graph if and only if there exists a proof π of* MLL *such that $S = [\![\pi]\!]$.*

The main issue with this criterion is the cost of its implementation. More specifically, if the net contains $n \in \mathbf{N}$ such $\parr$ links, then there are 2^n different switching graphs: this makes the complexity of this criterion exponential (in the number of links) in the worst case.

More efficient criteria, such as the *parsing* criterion of Banach [3], or the contractibility criterion [6], are defined through a rewriting system on nets. A net $\mathcal{H}$ is a proof net if and only if it can be rewritten to a net of a specific shape. Naively, these criteria operate in quadratic time, but may be optimized to run in linear time [14].

We will introduce in section IV a parsing criterion for second order multiplicative nets which coincides with that of Banach on MLL nets.

C. Generalised axioms

We introduce generalised axioms that, as in Ludics [15], we call *daimons* and denote $\maltese$. The daimon defines a sequent calculus rule with no premisses, which can introduce any sequent Γ: in particular, it generalises an axiom rules because it may introduce sequents of the form $A, A^\perp$. Considering this new rule we obtain new sequent calculus systems, $\mathsf{MLL}^{\maltese}$ which consists in the MLL system in which the axiom rule is substituted with the daimon rule. Similarly, we define $\mathsf{MLL}_2^{\maltese}$ as the MLL_2 system in which we replaced the axiom rule by the daimon rule.

Considering this new rule also gets us to consider new kinds of nets in which we don't allow axiom links but can contain daimon- ($\maltese$) links: these links have an empty source, but can have a target of any cardinality. A $\mathsf{MLL}^{\maltese}$ *nets* is a net which is a sum of link of $\mathcal{L}^{\maltese}_{\mathsf{MLL}}$ where:

$$\mathcal{L}^{\maltese}_{\mathsf{MLL}} = \left\{ \begin{array}{c} \langle p_1, p_2 \rhd_{\otimes} p_3\rangle,\ \langle p_1, p_2 \rhd_{\parr} p_3\rangle, \\ \langle p_1, p_2 \rhd_{\mathsf{cut}}\rangle,\ \langle \rhd_{\maltese} p_1, \dots, p_n\rangle \end{array} \middle| \begin{array}{c} n \in \mathbb{N}, \\ p_1, \dots, p_n \in \mathcal{P} \end{array} \right\}$$

Any MLL nets can canonically be mapped to $\mathsf{MLL}^{\maltese}$ nets by replacing each of its axioms $\langle \rhd_{\mathsf{ax}} p_1, p_2\rangle$ by a binary daimon link $\langle \rhd_{\maltese} p_1, p_2\rangle$. Conversely, any $\mathsf{MLL}^{\maltese}$ net which has only binary daimons (i.e. daimons of the form $\langle \rhd_{\maltese} p_1, p_2\rangle$) can be mapped to a MLL nets by replacing each binary daimon $\langle \rhd_{\maltese} p_1, p_2\rangle$ with an axiom $\langle \rhd_{\mathsf{ax}} p_1, p_2\rangle$.

The critera which tests the correctness of MLL nets can also test the correctness of $\mathsf{MLL}^{\maltese}$ nets, more precisely: applied to a $\mathsf{MLL}^{\maltese}$ net, the Danos-Regnier criterion and the parsing criterion characterise if the net S represents a proof π where π is a proof of $\mathsf{MLL}^{\maltese}$ (instead of simply MLL). In fact the use of daimons is closely related to the correctness criteria, they can be used to encode the partitions involved in the Danos-Regnier criterion [16]. Furthermore, they allow for a natural expression of the parsing rewriting: it consists simply of a rewriting of $\mathsf{MLL}^{\maltese}$ nets into $\mathsf{MLL}^{\maltese}$ nets. On the other hand, an MLL net will *never* rewrite into an MLL net after one step of parsing rewriting.

III. Second order nets

The first problem we must address is that of defining the $\mathsf{MLL}_2^{\maltese}$-nets: they will resemble the $\mathsf{MLL}^{\maltese}$-nets but will be equipped with new kind of links: the new connectives $\forall$ and $\exists$, but also, a new class of links that we call *pointers*. Before establishing the criterions we will also need to introduce the desequentialisation process, defining the net $[\![\pi]\!]$ associated with a proof–tree π from MLL_2.

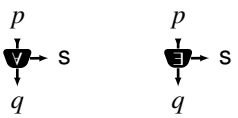
A. Adding pointers

In previous sections, the vertices of an hypergraph belonged to a set positions $\mathcal{P}$. In order to handle pointers we enrich the set of positions as $\mathcal{P} \cup \mathcal{P}^\bullet$ where the elements of $\mathcal{P}^\bullet$ are called *pointer ports* or pointer positions. We will usually denote pointer ports as $\mathsf{p}, \mathsf{q}, \mathsf{s}, \dots$. The elements of $\mathcal{P} \cup \mathcal{P}^\bullet$ are called *positions*, while elements of $\mathcal{P}$ will be called *regular positions*. Links in $\mathcal{L}^{\maltese}_{\mathsf{MLL}}$ will involve *only* regular positions.

We then consider two new kind of links, corresponding to the universal ($\forall$) and existential ($\exists$) quantifiers:

$$\mathcal{L}^{\maltese}_2 = \{\langle p \rhd_\forall q, \mathsf{s}\rangle, \langle p \rhd_\exists q, , \mathsf{s}\rangle \mid p, q \in \mathcal{P}, \mathsf{s} \in \mathcal{P}^\bullet\} \cup \mathcal{L}^{\maltese}_{\mathsf{MLL}}.$$

Links of the form $\langle p \rhd_\forall q, \mathsf{s}\rangle$ (resp.$\langle p \rhd_\exists q, \mathsf{s}\rangle$) are *universal links* (resp. *existential links*): these links contain both one target s which is a port position, and in fact are (up-to this point) the only links involving pointer positions. In illustrations, pointer positions are shown on the right of the links:



As explained in the introduction, these links are not sufficient to handle quantifiers properly. We need to introduce

a new class of links: *pointers links*. A *pointer* is a link with a single source position which is a pointer position, a single target position which is a regular position, and a label **p**. To avoid confusion such links will have a separate notation: we denote $\langle \mathtt{s} \blacktriangleright_{\mathbf{p}} t \rangle$ the pointer $\langle \mathtt{s} \rhd_{\mathbf{p}} t \rangle$. We now extend $\mathcal{L}_2^{✠}$ with the following types of pointers with forall-pointers and two types of existential pointers:

$$\mathrm{Ptr} = \{\langle \mathtt{s} \blacktriangleright_{\forall} t \rangle, \langle \mathtt{s} \blacktriangleright_{\exists}^{+} t \rangle, \langle \mathtt{s} \blacktriangleright_{\exists}^{-} t \rangle \mid \mathtt{s} \in \mathcal{P}^{\bullet}, t \in \mathcal{P}\}.$$

Suppose now given an hypergraph defined from links in $\mathcal{L}_2^{✠} \cup \mathrm{Ptr}$ such that each position is in the target set of exactly one $\mathcal{L}_2^{✠}$ link. Since the $\forall$ and $\exists$ links are the only (non pointers) links containing a port position, each pointer $\langle \mathtt{s} \blacktriangleright_{\mathbf{p}} t \rangle$ will have its source $\mathtt{s}$ belonging to the target of exactly one existential or universal link. The corresponding link will be called the *source link* of the pointer.

Definition 8. A *directed path* is a sequence of positions $p_1, \ldots, p_k$ such that for all $i = 1, \ldots, k-1$ there exists a non-pointer link $e \in \mathcal{L}_2^{✠}$ with $p_i \in |s(e)|$ and $p_{i+1} \in |t(e)|$.

A position p is said to be *above* a position q if there exists a directed path $p_1, \ldots, p_k$, $p_1 = p$ and $p_k = q$.

Definition 9. A MLL_2 *pre-net* is a sum $\mathcal{H}$ of links in $\mathcal{L}_2^{✠} \cup \mathrm{Ptr}$ such that:

- regular positions are in the source of at most one link,
- all position are in the target set of exactly one $\mathcal{L}_2$ link,
- source links of $\exists$ (resp. $\forall$) pointers are $\exists$ (resp. $\forall$) links,
- target positions of $\forall$ pointers are targets of ✠ links,
- target positions of $\exists$ pointers are *above* their source.

We distinguish universal and existential pointer links because the semantics of these pointers pointers are distinct, and as a consequence they will be treated differently in the correctness criterion. A positive existential (resp. negative existential) pointer is a pointer of the form $\langle \mathtt{s} \blacktriangleright_{\exists}^{+} t \rangle$ (resp. $\langle \mathtt{s} \blacktriangleright_{\exists}^{-} t \rangle$). In the typed case, this indicates the fact that the target of the link is a *positive* (resp. *negative*) occurence of the existential witness of the source $\exists$–link. We may write $\langle \mathtt{s} \blacktriangleright_{\exists} t \rangle$ to denote an existential pointer with an arbitrary sign. A universal pointer is a pointer of the form $\langle \mathtt{s} \blacktriangleright_{\forall} t \rangle$. In the typed case, this indicates the fact that the target position t depends on the variable quantified by the source $\forall$ link.

Remark 10. Note that in Definition 9, pointers have no restriction on their targets or sources. In particular, two pointers may have the same source: for instance $\langle \mathtt{s} \blacktriangleright_{\mathbf{p}} t \rangle$ and $\langle \mathtt{s} \blacktriangleright_{\mathbf{p}} t' \rangle$ can occur in the same pre–net. Similarly two distinct pointers may share their target, this may occur for instance when two distinct $\forall$ links point to the same position. In the typed case, this indicates that the position contains a formula in which both variables occur free. A $\forall$-pointer link may also share its target with an $\exists$-pointer link. In the typed case, this indicates that the existential witness in that position contains the variable.

Lastly, note that we do not disallow repetition of pointers: a net could contain two (or more) copies of the link $\langle \mathtt{s} \blacktriangleright_{\mathbf{p}} t \rangle$.

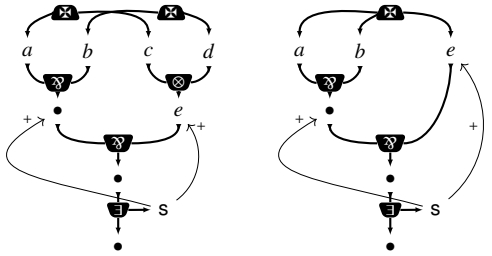


Fig. 4: Coherence of trees is not merely the isomorphism of the trees.

While this does not impact the results, we will suppose in the following that such a situation does not arise.

B. Existential pointers and coherence

One additional condition needs to be added to define *untyped* second-order nets. This aspect arises from our choice to work with untyped nets and has to do with existential quantifiers. More specifically: while existential pointers give the information of which sub-trees correspond to the *existential witnesses* of the $\exists$ rule, they do not ensure that these subtrees are *coherent*, i.e. that they can be understood as describing a single formula. For instance, consider the pre-nets of Figure 4, no sequent calculus proof can translate into the left-hand side pre–net: indeed, it requires the existential witness to be simultaneously of the form $A \otimes B$ and of the form $A \parr B$, which is not possible. A natural idea would be to ask the pointed subtrees to be isomorphic. But this is not compatible with the use of non-atomic axioms or daimon rules. Indeed, because daimon rules can introduce any sequent, the right-hand side pre-net of Figure 4 is the image of the following proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A, B, A \parr B}\,✠}{A \parr B, A \parr B}\,\parr}{(A \parr B) \parr (A \parr B)}\,\parr}{\exists X\ (X \parr X)}\,\exists$$

An obvious reason why the last existential rule is correct is that we have two occurrences of *the same* witness. In the typed case, this is guaranteed by the fact that this witness is the same formula (e.g. $A \parr B$ in the proof above). In the absence of explicit formulas, we use the notion of *coherence* of the subtrees above the target of existential pointers. Note that coherent subtrees are not only compatible but we also require that their pointed position match.

To properly state the definition, we first introduce a few notations. Given a pre-net and a binary relation $\sim$ on pointer positions, we say that positions p and p' are *existentially equivalent w.r.t.* $\sim$ when there exists a pointer link $\langle \mathtt{s} \blacktriangleright_{\exists}^{\star} p \rangle$ and a pointer link $\langle \mathtt{s}' \blacktriangleright_{\exists}^{\star} p' \rangle$ with $\mathtt{s} \sim \mathtt{s}'$ (here $\star \in \{+, -\}$). Given a pre-net and a position p, we write $\forall_{\uparrow}(p)$ the set of sources of pointer links whose target is above p. Given

a binary relation $\sim$ on pointer positions, we write $\forall_\uparrow(p) \sim \forall_\uparrow(p')$ when there exists a one-to-one mapping $f : \forall_\uparrow(p) \to \forall_\uparrow(p')$ such that $f(x) \sim x$.

Given a binary relation $R \subset X \times X$ and two elements $x, x' \in X$, we define $R_{x \leftrightarrow x'}$ as follows:

$$R_{x \leftrightarrow x'} = (R \cup \{(x, x'), (x', x)\}) \setminus \{(x, x), (x', x')\}$$

Definition 11. Let N be a pre-net $(S, \mathbf{P})$, p and p' be two positions, and $\sim$ be a binary relation on positions. We say p and p' are *coherent* w.r.t $\sim$ when: either p and p' are existentially coherent w.r.t. $\sim$, or they are not target of existential pointer links and satisfy one of the following conditions:

- p or p' is the target of a daimon link, and $\forall_\uparrow(p) \sim \forall_\uparrow(p')$;
- p and p' are both conclusions of $\otimes$ links (resp. $\parr$ links), whose hypotheses are pairwise coherent w.r.t. $\sim$;
- p and p' are both conclusions of $\forall$ links (resp. $\exists$ links) of pointer position $\mathtt{s}$ and $\mathtt{s}'$, whose hypotheses are coherent w.r.t. the relation $\sim_{\mathtt{s} \leftrightarrow \mathtt{s}'}$.

We say that p and p' are *coherent*, written $p \frown p'$ when they are coherent w.r.t. the identity on pointer positions (i.e. the binary relation $x \sim x' \Leftrightarrow x = x'$).

We define in a similar way the notion of anticoherent positions. Given a pre-net, and a binary relation $\sim$ on pointer positions, we say that positions p and p' are existentially anticoherent w.r.t. $\sim$ if there exists pointer positions $\mathtt{s} \sim \mathtt{s}'$ and:

- either there exists links $\langle \mathtt{s} \blacktriangleright_\exists p \rangle$ and $\langle \mathtt{s}' \blacktriangleright_\forall p' \rangle$,
- or there exists links $\langle \mathtt{s} \blacktriangleright_\forall p \rangle$ and $\langle \mathtt{s}' \blacktriangleright_\exists p' \rangle$.

Definition 12. Let N be a pre-net $(S, \mathbf{P})$, p and p' be two positions, and $\sim$ be a binary relation on positions. We say p and p' are *anti-coherent* w.r.t $\sim$ when p and p' are existentially anticoherent w.r.t. $\sim$, or they are not targets of pointer links and:

- p or p' is the target of a daimon link, and $\forall_\uparrow(p) = \forall_\uparrow(p')$;
- p and p' are conclusions of $\otimes$ and $\parr$ links, whose hypotheses are pairwise anticoherent w.r.t. $\sim$;
- p and p' are conclusions of $\forall$ link and $\exists$ links of pointer position $\mathtt{s}$ and $\mathtt{s}'$, whose hypotheses are anticoherent w.r.t. the relation $\sim_{\mathtt{s} \leftrightarrow \mathtt{s}'}$.

We say that p and p' are *anticoherent*, written $p \asymp p'$ when they are anticoherent w.r.t. the identity on pointer positions (i.e. the binary relation $x \sim x' \Leftrightarrow x = x'$).

Definition 13. A pre-net S is a *net* if for all existential link $\langle p \rhd_\exists q, \mathtt{s} \rangle$ in S and all positions p, p':

- if there are pointer links $\langle \mathtt{s} \blacktriangleright^+_\exists p \rangle$ and $\langle \mathtt{s} \blacktriangleright^+_\exists p' \rangle$ (resp. $\langle \mathtt{s} \blacktriangleright^-_\exists p \rangle$ and $\langle \mathtt{s} \blacktriangleright^-_\exists p' \rangle$), then $p \frown p'$ in the net without these pointer links;
- if there are pointer links $\langle \mathtt{s} \blacktriangleright^-_\exists p \rangle$ and $\langle \mathtt{s} \blacktriangleright^+_\exists p' \rangle$ (resp. $\langle \mathtt{s} \blacktriangleright^+_\exists p \rangle$ and $\langle \mathtt{s} \blacktriangleright^-_\exists p' \rangle$), then $p \asymp p'$ in the net without these pointer links.

With some work, one can show that for each $\exists$ link in a net, one can reconstruct a minimal witness, which is intuitively the union of the subtrees above targets of existential pointers (or their dual, depending on the polarity of the pointer). Coherence (and anticonherence) insures that taking such a union is possible.

C. Pre-soundness

We can associate a proof π from the sequent calculus of $\mathsf{MLL}_2^{\maltese}$ with a $\mathsf{MLL}_2^{\maltese}$ net, that we denote $[\![\pi]\!]$, which is defined by induction on the proof tree π. The inductive cases of the $\otimes$- $\parr$- and cut- rules are as in the multiplicative case [13], with the additional presence of pointers but in these rules those do not play any role and are just carried through the induction unchanged. The cases of the $\forall$- and $\exists$- connectives is the novelty here, we add a link at below the right position but more importantly we may add pointers, it is important to understand what are the targets of the pointers which have their source in the new quantifier link. To handle universal pointers we assume the existence of an injection $X \mapsto \mathtt{s}_X$ mapping propositional variables to port positions.

Definition 14. Given a finite set of propositional variables $\mathcal{V}$, the *representation with traced variables* $\mathcal{V}$ of a proof π in $\mathsf{MLL}_2^{\maltese}$ is the net $\langle \pi; \mathcal{V} \rangle$ defined by induction in Figure 5. In the base case i.e. π is a proof made of a single daimon rule, a pointer $\langle \mathtt{s}_X \blacktriangleright_\forall p_i \rangle$ is added if and only if the variable X belongs to V and occurs free in the formula A_i. In the case of the $\exists$-rule the position $w_1, \dots, w_l$ are the targets of the new existential pointers and $\vec{w}$ is an enumeration of the positions at address $\xi \in \{\mathrm{l}, \mathrm{r}, \mathrm{up}\}^*$ with respect to p in the net S (l indicates to go up on the left of p, and so on $\dots$) so that ξ is the address of an occurence of Y in A (see Figure 5), i.e. a w_i is a position at the address ξ relatively to p where ξ is the address of a Y's occurrences in A. In the case of the $\forall$-rule no new pointer is added simply the we add a link $\langle p \rhd_\forall q, \mathtt{s}_X \rangle$ and the pointers of source $\mathtt{s}_X$ naturally become the pointers of the added link and their targets are $v_1, \dots, v_l$ in Figure 5 (note that universal-pointers are carried along the induction defining $[\![\pi]\!]$).

The *(canonical) representation* of a proof π is $\langle \pi; \mathcal{V}_P \rangle$ where $\mathcal{V}_P$ is the set of universal variables of π, we denote it $[\![\pi]\!]$.

Remark 15. The function $\langle \cdot; \mathcal{V} \rangle$ associates with each available formula occurence in a proof π a position in the hypergraph $\langle \pi; \mathcal{V} \rangle$; this map is denoted pos_π and is bijective, i.e. any position in $\langle \pi; \mathcal{V} \rangle$ is associated with a unique available occurence of a formula in π.

In the exists rule of MLL_2 (Figure 3) the occurence of the formula B in $A[X \leftarrow B]$ are the *witnesses occurence* of the rule; the occurence of X in A are the *positive witnesses occurence*, the occurence of $X^\perp$ in A are the *negative witnesses occurence*. A proof tree is *complete* if for each of its exists rule r the witnesses occurence of r are accessible occurences (section II). The next proposition explains that a proof-tree of $\mathsf{MLL}_2^{\maltese}$ or MLL_2 can always be extended so that it is complete; for MLL_2 an obvious solution is to eta-expand the proof, although it is not necessary to fully eta-expand it and use only atomic axioms, we merely need that the

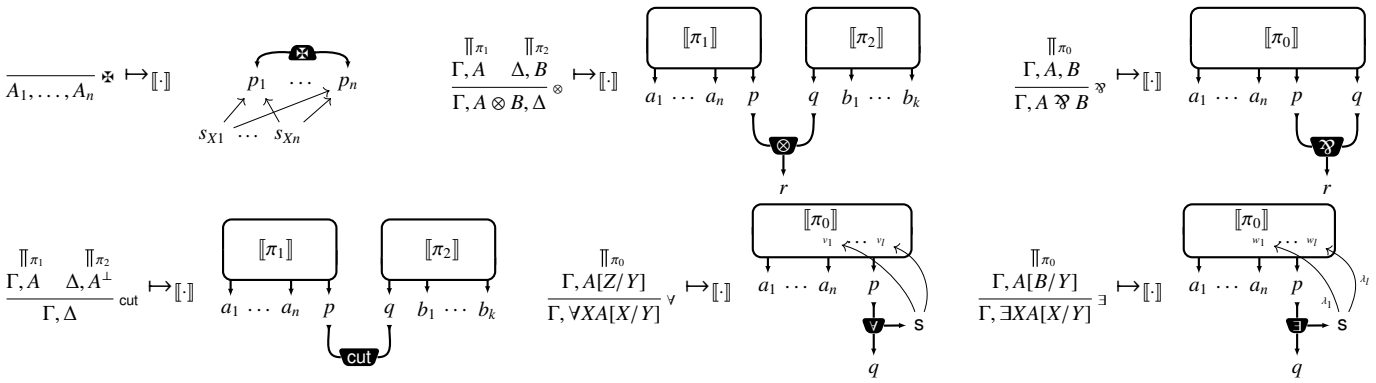


Fig. 5: The desequentialisation process: it associates with every proof of the sequent calculus MLL_2 (and more generally $\mathsf{MLL}_2^{\maltese}$) a $\mathsf{MLL}_2^{\maltese}$ net.

witness occurences become available, i.e. principal in some rule. In fact, to obtain a complete proof the eta expansion of quantifiers will never be needed, because an exists cannot have for witness a (strict) subformula of $\forall XA$ or of $\exists XA$ so only the eta expansion of $\otimes$- and $\parr$- connectives is needed.

Proposition 16. *For any π proof tree of MLL_2 (resp. $\mathsf{MLL}_2^{\maltese}$) there exists a complete proof tree π' such that π eta expands (resp. proof–search expands, see Figure 6) to π', i.e. $\pi \to_\eta \pi'$ (resp. $\pi \to_s \pi'$).*

Proposition 17. *For each proof π of $\mathsf{MLL}_2^{\maltese}$, $[\![\pi]\!]$ is a net.*

Proof. It is straightforward to show that $[\![\pi]\!]$ is a pre-net, then one shows that $[\![\pi]\!]$ has coherent witness trees: this is because the (roof of) trees which correspond to a same witness are associated with occurences of the same formula (i.e. the formula B in the premisse of the $\exists$ rule $\Gamma, A[B/X]$). □

Remark 18. Proposition 17 shows that proofs desequentialise to nets, however not all nets are the desequentialisation of a proof. As in the multiplicative case nets which contain switching cycles or switching disconnection don't represent any proof, the novelty is that some nets $(S, \mathbf{P})$ may be multiplicatively correct (i.e. the switching graphs of S are acyclic and connected) but still be incorrect. For instance two existential pointers may be conflicting, more precisely when the paths of these pointers (i.e. the path from the source to the target of the pointer) intersect and belong to a same unique descending path (see Figure 7). Another case is when a forall rule is performed when the variable X is not free in the context (such as the net $S_{\overline{\eta}}$ of section I whose untyped version is right hand side net of Figure 7), this will become clear when defining the parsing reduction steps.

Remark 19. The nets we have introduced can represent proof (Proposition 17) furthermore two proofs which are equivalent upto some rules-commutations are mapped to the same net (up-to isomorphism). For instance, two proofs which are equivalent for the $(\forall/\forall)$ commutations (see below, where X and Y do not occur free in Γ) will be mapped to the same net.

$$\dfrac{\dfrac{\llbracket\pi}{A, B, \Gamma}}{\dfrac{\forall XA, B, \Gamma}{\forall XA, \forall YB, \Gamma}\,\forall}\,\forall \quad\sim\quad \dfrac{\dfrac{\llbracket\pi}{A, B, \Gamma}}{\dfrac{A, \forall YB, \Gamma}{\forall XA, \forall YB, \Gamma}\,\forall}\,\forall$$

We can capture proof equivalences by isomorphism because of the absence of *boxes* (by contrast, boxes are involved in the works [1] and [11]). Because of that, our representation of proofs as nets captures all natural commutations.

IV. Second order proof nets

We will define a reduction procedure on $\mathsf{MLL}_2^{\maltese}$ nets that will allow us to define the notion of "proof net". However, the reduction rules require to keep track of some additional dependencies during the parsing rewriting. As a consequence, we will introduce new pointer links, which will be used *only* for the purpose of rewriting. We denote the class of these additional pointers Ptr_C:

$$\mathrm{Ptr}_C = \{\langle \mathsf{s} \dashrightarrow_\exists q\rangle, \langle \mathsf{s} \dashrightarrow_\forall q\rangle, \langle \mathsf{s} \rightsquigarrow_\forall \mathsf{s}'\rangle \mid \mathsf{s}, \mathsf{s}' \in \mathcal{P}^\bullet, q \in \mathcal{P}\}.$$

Pointers of the form $\langle \mathsf{s} \dashrightarrow_\exists q\rangle$ (resp. $\langle \mathsf{s} \dashrightarrow_\forall q\rangle$, resp. $\langle \mathsf{s} \rightsquigarrow_\forall \mathsf{s}'\rangle$) are called *existential ghosts* (resp. *universal ghosts*, resp. *dependency pointers*).

In the following, we will therefore work with a pair of a $\mathsf{MLL}_2^{\maltese}$-net, together with a *correction pointer structure*.

Definition 20. Let S be a $\mathsf{MLL}_2^{\maltese}$ net. A *correction pointer structure for S* is a sum $\mathbf{P}$ of links from Ptr_C whose source and target are positions in S and such that:

- universal ghosts are subject to the same constraints as universal pointers: their source is a pointer positions which is the target of a $\forall$ link, and their target is the target of a $\maltese$ link;
- existential ghosts are subject to the same constraints as existential pointers: their source is a pointer positions which is the target of a $\exists$ link, and their target is a position *above* their source;
- dependency pointers are pointers from universal links to existential links: is a pointer positions which is the

$$\frac{}{\Gamma, A \parr B}\,\text{✠} \to_s \frac{\dfrac{\Gamma, A, B}{\Gamma, A \parr B}\,\parr}{}\,\text{✠} \qquad \frac{}{\Gamma, A \otimes B, \Delta}\,\text{✠} \to_s \frac{\dfrac{\Gamma, A}{\Gamma, A \otimes B, \Delta}\,\otimes \quad \dfrac{\Delta, B}{}}{}\,\text{✠} \qquad \frac{}{\Gamma, \forall X A}\,\text{✠} \to_s \frac{\dfrac{\Gamma, A[Y/X]}{\Gamma, \forall X A}\,\forall}{}\,\text{✠} \qquad \frac{}{\Gamma, \exists X A}\,\text{✠} \to_s \frac{\dfrac{\Gamma, A[B/X]}{\Gamma, \exists X A}\,\exists}{}\,\text{✠}$$

Fig. 6: Proof search rewriting in $\mathsf{MLL}_2^{\text{✠}}$ sequent calculus. In the forall-rule the eigenvariable Y is fresh.

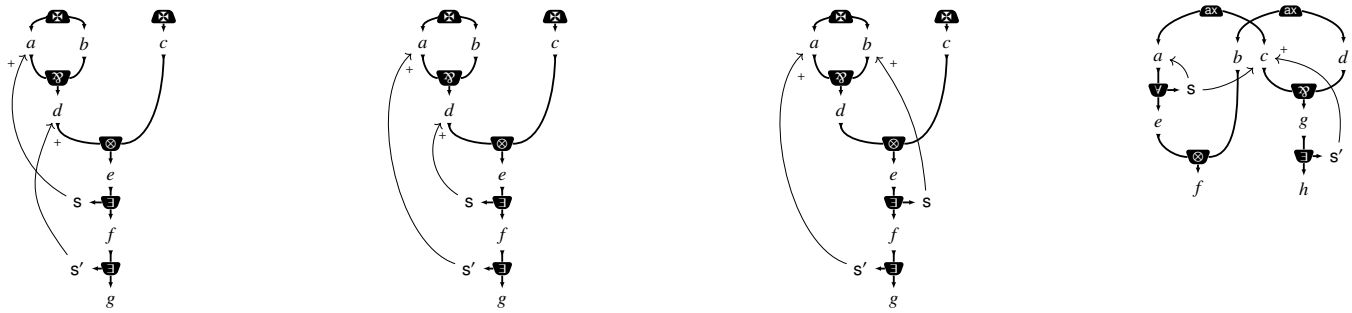


Fig. 7: Interaction of existential pointers. The first two nets are incorrect because the existential pointers will create a conflict (an exists-pointer and an exists-ghost will share the same target during the parsing reduction), the third net is correct, the existential pointers will not conflict: they will only meet when they are both ghosts. The right-most net is incorrect because the pointers of the forall-link cannot become ghost, the parsing rewriting will not be able to perform any step that is not a rerouting.

target of a ∀ link, and their target is a pointer positions which is the target of a ∃ link.

Definition 21. A *correction net* is a pair $(S, \mathbf{P})$ of a net S and a correction pointer structure for S, we will denote such pairs $S \bullet \mathbf{P}$. Given a correction net $S \bullet \mathbf{P}$, a position p is *active* when p belongs to the target set of a ✠ link of S, and such that if p is the target of an existential pointer, then it is *not* the target of another existential pointer, nor the target of an existential ghost.

An active position is *existential-free* if it is not the target of an existential pointer.

A (non-pointer) link is *active* if its source set contains only active positions. It is *existential-free* if all its source positions are existential-free.

Notation 22. Given a $\mathsf{MLL}_2^{\text{✠}}$ net S we denote by $S(\mathsf{s}\!\rightarrow)$ the pointer links of source s in S, i.e. the pointers of the form $\langle \mathsf{s} \blacktriangleright t \rangle$ for some position t. Given a position p, $S(\rightarrow p)$ denotes the pointers of target p in S i.e. of the form $\langle \mathsf{s} \blacktriangleright p \rangle$ for some port position s. Similarly we define the notations for a correction pointer structure $\mathbf{P}$ and define the notation for the ghosts $\mathbf{P}(\mathsf{s}\dashrightarrow)$.

Given a correction pointer structure $\mathbf{P}$ we define $\mathbf{P}(\rightsquigarrow \mathsf{s})$ the dependency pointers of target s, i.e. of the form $\langle \mathsf{s}' \rightsquigarrow_\forall \mathsf{s} \rangle$. Given a universal pointer $\langle \mathsf{s} \blacktriangleright_\forall p \rangle$ the dependency substitution $\langle \mathsf{s} \blacktriangleright_\forall p \rangle [p \leftsquigarrow \mathsf{s}']$ returns $\langle \mathsf{s} \rightsquigarrow_\forall \mathsf{s}' \rangle$. Given a dependency pointer $\langle \mathsf{s} \rightsquigarrow_\forall \mathsf{s}' \rangle$ the ghost substitution $\langle \mathsf{s} \rightsquigarrow_\forall \mathsf{s}' \rangle [\mathsf{s}' \dashleftarrow r]$ returns $\langle \mathsf{s} \dashrightarrow_\forall r \rangle$. Given an existential pointer $\langle \mathsf{s} \blacktriangleright_\exists p \rangle$ the ghost substitution $\langle \mathsf{s} \blacktriangleright_\exists p \rangle [p \dashleftarrow r]$ returns $\langle \mathsf{s} \dashrightarrow_\exists r \rangle$. These substitutions naturally can be lifted to sets of pointer links.

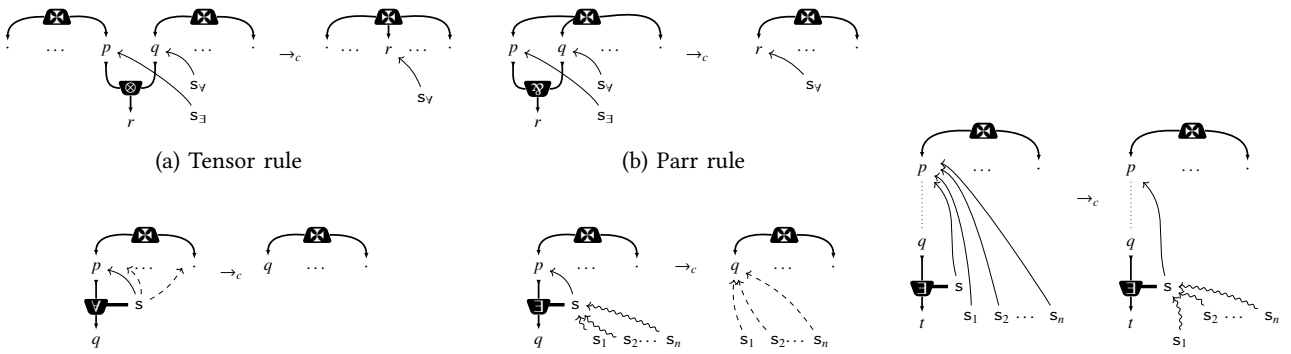
Notation 23. A *context* for a net S is an hypergraph $\mathbf{C}$ such that the sum $\mathbf{C} + S$ is a net, then that sum is denoted $\mathbf{C}[S]$ (in particular $\mathbf{C}$ can contain pointers whose target lies within the positions of S). The substitution $[p \leftarrow q]$ is a map on positions and maps p to q while mapping any other position a to itself a. Substitutions can be lifted to sets in the usual way. Substitutions can also be lifted single-link hypergraphs such that $\mathrm{t}(e[p \leftarrow q]) = \mathrm{t}(e)[p \leftarrow q]$ while $\mathrm{s}(e[p \leftarrow q]) = \mathrm{s}(e)[p \leftarrow q]$. Substitutions can therefore be lifted to nets. We will denote by $\mathbf{C}_\theta[S]$ the hypergraph $\theta\mathbf{C} + S$.

We define the *parsing reduction* on correction nets as a labelled rewriting $S \bullet \mathbf{P} \xrightarrow{e}_{\mathsf{P}} S' \bullet \mathbf{P}'$ on active links. The rewriting rules are shown in Figure 9 and are subject to the following constraints.

- Par rule: it applies on an existential-free active ⅋ link, such that the two positions in the source set are targets of the same ✠ link. This contracts the ⅋-link into a single position r (the conclusion of the link). Pointers to the sources of the initial ⅋-link are rerouted to r.
- Tensor rule: it applies on an existential-free active ⊗ link, such that the two positions in the source set are targets of distinct ✠ links. This contracts the ⊗ link into a single position r (the conclusion of the link), merging the two ✠ links. Pointers to the sources of the initial ⊗ link are rerouted to r.
- Forall rule: it applies on an existential-free active ∀ link

$$\langle \rhd_{\text{✠}}\ p_1, \ldots, p_n, p \rangle + \langle p \rhd_\forall r, \mathsf{s} \rangle$$

such that there is at most one universal pointer from s and it can only go to p and all universal ghosts of source s have their target among $p_1, \ldots, p_n$. This contracts the ∀ link into a single position r. Pointers of source s are deleted, and pointers of target p are rerouted to r.
- Exists rule: it applies on an existential-free active ∃ link $\langle p \rhd_\exists r, \mathsf{s} \rangle$. This contracts the ∃ link into a single



(a) Tensor rule

(b) Parr rule

(c) Universal rule (add inactive pointers)

(d) Existential rule (existential pointer is inactive, may be other inactive existential pointers)

(e) Pointers rule (the existential pointer is the only one)

Fig. 8: Parsing rewriting rules illustrated

position r. Pointers of source $\mathtt{s}$ are deleted, dependency pointers of target $\mathtt{s}$ are replaced by universal ghosts of target r, and pointers of target p are rerouted to r.

- Pointers rule: it is the only rule that applies to an active position

$$\langle \rhd_{\maltese}\ p_1, \ldots, p_n, p\rangle + \langle \mathtt{s} \blacktriangleright_{\exists} p\rangle$$

which is not existential-free. This replaces the unique existential link $\langle \mathtt{s} \blacktriangleright_{\exists} p\rangle$ by an existential ghost $\langle \mathtt{s} \dashrightarrow_{\exists} p\rangle$, and universal links of target p are replaced by dependency links to $\mathtt{s}$.

Definition 24. Given an integer n we denote $\maltese_n$ the net consisting of a single daimon link with n conclusions, i.e. $\langle \rhd_{\maltese}\ p_1, \ldots, p_n\rangle$. A *proof-net* is a net S that reduces to a daimon i.e. such that $S \to_{\mathsf{P}} \maltese_n$ for some integer n.

V. Soundness

Theorem 25 (Soundness)**.** *The image of a sequent calculus proof is a proof net.*

Theorem 25 is obtained by proving a more general result which is better suited for inductive reasoning. Recall that the representation of a proof $\langle \pi; \mathcal{V}\rangle$ naturally comes with a localising map pos_π mapping each active formulas in π to a position in $\langle \pi; \mathcal{V}\rangle$. In fact, more precisely, given $A_1, \ldots, A_n$ the conclusion sequent of π the positions $\mathsf{pos}_\pi(A_1), \ldots, \mathsf{pos}_\pi(A_n)$ are the conclusions of the net $\langle \pi; \mathcal{V}\rangle$.

Lemma 26. *Given a sequent calculus proof π of conclusion $A_1, \ldots, A_n$, $\langle \pi; \mathcal{V}\rangle$ reduces to $\maltese_n + \mathbf{P}$ where $\mathbf{P}$ contains universal pointers $\langle \mathtt{s}_X \blacktriangleright_\forall \mathsf{pos}_\pi(A_i)\rangle$ when X belongs to $\mathcal{V}$ and X occurs free in A_i.*

The induction is more or less straightforward, except for the $\exists$ quantifier rule case, which requires the following proposition to ensure that no ill-situation arise between existential pointers (such as the one described in Remark 18). Say that a position is *bound* if it is above an exists pointer or if it is in the descending path from an exists pointer to its link. A position is *free* when it is not bound. Then the following is true.

Proposition 27. *Given a net S and e terminal existential link for S such that e points only to free positions in S; if $S \to_{\mathsf{P}} S'$ then $S + e \to_{\mathsf{P}} S' + e$.*

One uses the previous proposition in order to prove Lemma 26, specifically, when the proof π ends with a terminal $\exists$- rule. In that case $\langle \pi; \mathcal{V}\rangle$ will be mapped to $\langle \pi_0; \mathcal{V}\rangle + e + \mathbf{P}_e$ where π_0 is the immediate subproof in π and e is the new existantial link while $\mathbf{P}_e$ are the added pointers. In order to apply Proposition 27, one must observe that e is indeed a terminal link for $\langle \pi_0; \mathcal{V}\rangle$ and that the new pointers $\mathbf{P}_e$ points to free positions in $\langle \pi_0; \mathcal{V}\rangle$ (this is ensured by the types in the sequent calculus).

VI. Completeness

In this section, we prove:

Theorem 28. *A proof net is the image of a sequent calculus proof.*

Since we are working with untyped nets, we are working up to an implicit universal quantification. Indeed, a given proof net represents many different sequent calculus proofs that share the same general structure (derivation tree) but differ in the formulas introduced in the axioms.

This appeared in the proof of soundness, but did not induce additional complications. This will however need to be tackled properly in the proof of completeness. Indeed, the proof will show that one can reconstruct a sequent calculus proof from a proof structure and a parsing reduction to $\maltese_k$. Some information about the formulas involved will only be known after several steps in this reconstruction. We therefore need to proceed inductively on families of proofs sharing some structure (e.g. proofs that end with a tensor rule in which the left formula depends on a given variable X). Those are abstracted by the notion of *scheme*.

$$\begin{array}{rcl}
\mathbf{C}[\langle \rhd_{\bar{\pi}} \vec{p}, p\rangle + \langle \rhd_{\bar{\pi}} \vec{q}, q\rangle + \langle p, q \rhd_{\otimes} r\rangle] \bullet \mathbf{P} & \to_{\mathsf{P}} & \mathbf{C}_{r\leftarrow p,q}[\langle \rhd_{\bar{\pi}} \vec{p}, \vec{q}, r\rangle] \bullet \mathbf{P}[r \leftarrow p, q] \\
\mathbf{C}[\langle \rhd_{\bar{\pi}} \vec{p}, p, q\rangle + \langle p, q \rhd_{\parr} r\rangle] \bullet \mathbf{P} & \to_{\mathsf{P}} & \mathbf{C}_{r\leftarrow p,q}[\langle \rhd_{\bar{\pi}} \vec{p}, r\rangle] \bullet \mathbf{P}[r \leftarrow p, q] \\
\mathbf{C}[\langle \rhd_{\bar{\pi}} \vec{p}, p\rangle + \langle p \rhd_{\vee} r, \mathsf{s}\rangle + S(\mathsf{s}\to)] \bullet \mathbf{P} + \mathbf{P}(\mathsf{s}\dashrightarrow) & \to_{\mathsf{P}} & \mathbf{C}_{r\leftarrow p}[\langle \rhd_{\bar{\pi}} \vec{p}, r\rangle] \bullet \mathbf{P}[r \leftarrow p] \\
\mathbf{C}[\langle \rhd_{\bar{\pi}} \vec{p}, p\rangle + \langle p \rhd_{\exists} r, s\rangle] \bullet \mathbf{P} + \mathbf{P}(\mathsf{s}\dashrightarrow) + \mathbf{P}(\rightsquigarrow \mathsf{s}) & \to_{\mathsf{P}} & \mathbf{C}_{r\leftarrow p}[\langle \rhd_{\bar{\pi}} \vec{p}, r\rangle] \bullet \mathbf{P}[r \leftarrow p] + \mathbf{P}(\rightsquigarrow \mathsf{s})\,[\mathsf{s} \dashleftarrow r] \\
\mathbf{C}[\langle \rhd_{\bar{\pi}} \vec{p}, p\rangle + \langle s \blacktriangleright_{\exists} p\rangle + S(\to p)_{\vee}] \bullet \mathbf{P} & \to_{\mathsf{P}} & \mathbf{C}[\langle \rhd_{\bar{\pi}} \vec{p}, p\rangle] \bullet \mathbf{P} + \langle s \dashrightarrow_{\exists} p\rangle + S(\to p)_{\vee}\,[p \rightsquigarrow \mathsf{s}]
\end{array}$$

Fig. 9: The parsing criterion: rewriting rules.

A. Schemes

Definition 29 (Schemes)**.** We consider a new set of variables $\mathrm{Var}^{(2)}$ disjoint from Var. We define *schemes* inductively as follows:

$$S := F(X_1, \dots, X_n) \mid X \mid S \otimes S \mid S \parr S \mid \exists X\ S \mid \forall X\ S,$$

where $X, X_1, \dots, X_n \in \mathrm{Var}$, and $F \in \mathrm{Var}^{(2)}$.

Schemes are a generalisation of MLL_2 formulas, they can be instantiated. Schemes impose some contraints on variables: we expect variables in $\mathrm{Var}^{(2)}$ to abstract formulas of MLL^2 that contain some specific variables in Var and that do not contain some other specific variables in Var. This translates through the notion of *explicit variables*: those are the variables for which constraints have to be satisfied.

Definition 30. An *instantiation with explicit variables* $B \subseteq \mathrm{Var}$ is function σ mapping schemes to MLL_2 formulas respecting the following induction:

- $\sigma(X)$ is X.
- $\sigma(F(X_1, \dots, X_n))$ is a formula such that for all i $X_i \in \mathrm{FV}(A)$, and for all $Y \in B \setminus \{X_1, \dots, X_n\}$, $Y \notin \mathrm{FV}(A)$;
- $\sigma(S_1 \square S_2) = \sigma(S_1) \square \sigma(S_2)$; for any binary connective.
- $\sigma(\forall X S) = \forall X \sigma S$ and $\sigma(\exists X S) = \exists X \sigma S$.

A formula A is an *instance with explicit variables* B of a scheme S if there exists a instantiation σ with explicit variables B such that $A = \sigma S$.

Example 31. Consider the scheme $S = \forall X F(X) \otimes F(Y)$. Then, in the empty set of explicit variables, we have that:

- $S \vDash_{\emptyset} \forall X (X \otimes Y) \otimes (X \otimes Y)$;
- $S \nvDash_{\emptyset} \forall X X \otimes (X \otimes Z)$, because a same second order propositional variable F must be instantiated by a same formula
- $S \nvDash_{\emptyset} \forall X X \otimes (Y \parr Z)$, because X is an explicit variable yet it appears in the instance of $F(Y, Z)$;
- $S \nvDash_{\emptyset} \forall X X \otimes (Y \parr Z)$, because the instance of $F'(W)$ does not contain the variable W;
- $S \nvDash_{\emptyset} \forall X X \otimes W$, because F is instanced to two distinct formulas X and W.

We will also need to keep track of existentially and universally quantified variables. We define the set $\mathrm{BV}(S)$ of bounded variables of a scheme S inductively:

- $\mathrm{BV}(F(X_1, \dots, X_n)) = \emptyset$;
- $\mathrm{BV}(S_1 \otimes S_2) = \mathrm{BV}(S_1) \cup \mathrm{BV}(S_2)$;
- $\mathrm{BV}(S_1 \parr S_2) = \mathrm{BV}(S_1) \cup \mathrm{BV}(S_2)$;
- $\mathrm{BV}(\exists X, S) = \mathrm{BV}(S) \cup \{X\}$;
- $\mathrm{BV}(\forall X, S) = \mathrm{BV}(S) \cup \{X\}$.

Definition 32. A *sequent of schemes* (or scheme of sequent) is a sequence of schemes $\vdash S_1, \dots, S_n$. We define $\mathrm{BV}(\vdash S_1, \dots, S_n)$ as the union of the sets $\mathrm{BV}(S_1), \dots, \mathrm{BV}(S_n)$.

A sequent of schemes $\vdash S_1, \dots, S_n$ can also be instantiated.

Definition 33. An instantiation σ with explicit variables B is *compatible* with a sequent of scheme $S_1, \dots, S_n$ if $\mathrm{BV}(S_1, \dots, S_n) = B$. A sequent $\vdash A_1, \dots, A_n$ is an *instantiation* of a sequent of schemes $S_1, \dots, S_n$ if there exists an instantiation compatible with $S_1, \dots, S_n$ such that $\vdash \sigma(S_1), \dots, \sigma(S_n)$ equals $\vdash A_1, \dots, A_n$.

Proofs can also be defined for schemes as in Figure 10. The proof of a scheme π instantiates to a proof π' of $\mathsf{MLL}_2^{\bar{\pi}}$ whenever there exists an instantion σ with explicit variable which corresponds to the variables in π, such that $\sigma(\pi)$ equals π'. Furthermore, as $\mathsf{MLL}_2^{\bar{\pi}}$ proofs, a proof of schemes π can be mapped to a net $\pi^\dagger$ as illustrated in Figure 17. In fact, the representation of a proof of schemes π and the representation of its instances are the same (Proposition 36).

To establish Theorem 28, we in fact prove the following technical result.

Lemma 34 (Technical lemma)**.** *Given a proof net P, there exists a scheme of sequent $\vdash S_1, \dots, S_n$ and a proof π of S (in $\mathsf{MLL}_2^{\bar{\pi}\dagger}$) such that $\pi^\dagger = P$.*

The proof of this follows from several key lemmas, stated in subsection VI-B, as well as the following proposition that relates proofs of schemes with proofs of $\mathsf{MLL}_2^{\bar{\pi}}$.

Proposition 35. *If $\vdash S_1, \dots, S_n$ is a scheme of sequent and π is a proof of $\vdash S_1, \dots, S_n$ (in $\mathsf{MLL}_2^{\bar{\pi}\dagger}$), then for each instantiation σ of $\vdash S_1, \dots, S_n$: $\sigma(\pi)$ is a proof of $\vdash \sigma(A_1), \dots, \sigma(A_n)$ in $\mathsf{MLL}_2^{\bar{\pi}}$.*

Proof. One checks that the rules of $\mathsf{MLL}_2^{\bar{\pi}\dagger}$ are mapped through σ to rules of $\mathsf{MLL}_2^{\bar{\pi}}$. Then, by induction one easily constructs a proof tree in $\mathsf{MLL}_2^{\bar{\pi}}$ from a proof tree π in $\mathsf{MLL}_2^{\bar{\pi}\dagger}$. □

Proposition 36. *Given a proof of schemes π, and an instantiation σ with explicit variables B; the proof structures $\langle \pi; B\rangle^\dagger$ and $\langle \sigma(\pi); B\rangle$ are equal.*

We are now ready to prove the completeness theorem stated at the beginning of the section.

Proof of Theorem 28. Let S be a proof net, using the technical result Lemma 34, there exists a scheme calculus proof π such that $S = [\![\pi]\!]^\dagger$. π is a scheme calculus proof tree of conclusion $S_1, \dots, S_n$ hence for any instantiation σ of $S_1, \dots, S_n$ the

proof tree $\sigma(\pi)$ is a proof of $\mathsf{MLL}_2^{\Uparrow}$ (Proposition 35). Recall that $[\![\pi]\!]^\dagger = \langle \pi; V\rangle^\dagger$ where V are the universal variables of π; by Proposition 36 $\langle \pi; V\rangle^\dagger$ and $\langle \sigma(\pi); V\rangle$ are equal. Finally since the variables of π and $\sigma(\pi)$ are the same $\langle \sigma(\pi); V\rangle$ is $[\![\pi]\!]$. We conclude that $S = [\![\sigma(\pi)]\!]$. □

B. Proof of the technical lemma

Notation 37. Given a net S and a correction net $S' \bullet \mathbf{P}$ we denote $S \approx S' \bullet \mathbf{P}$ whenever $S = S'$ i.e. the underlying net of the correction net $S' \bullet \mathbf{P}$ is S.

Proposition 35 trivially implies a more general result which applies to correction nets rather than nets.

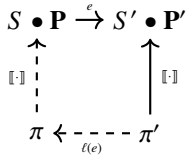
Corollary 38. *Given a scheme proof π and σ an instantiation for π whenever $S \bullet \mathbf{P} \approx [\![\pi]\!]^\dagger$ then $S \bullet \mathbf{P} \approx [\![\sigma(\pi)]\!]$.*

Given a proof-tree π and d one of its daimon rule given a proof-tree π_0 we denote $\pi[\pi_0/d]$ the proof obtained by replacing the daimon d in π with the proof π_0. The following lemma (Lemma 40) involves the *proof search rewriting* $\to_s$: it rewrites a proof-tree consisting of a single daimon rule into a proof tree containing a single rule, this rewriting then generalises to proof trees by the contextual closure in the straightforward way, if $\pi \to_s \pi'$ then for any proof ρ, $\rho[\pi/d] \to_s \rho[\pi/d]$. In particular one can easily see that each rewriting steps defining $\to_s$ introduce a correct rule or applies an existential subsitution (this preserves correctness), furthermore one can observe that any proof tree can be the result of a proof search starting from a proof made of a single daimon link, this result in the following proposition:

Proposition 39. *π is a proof tree, if and only if, there exists* Dai *a proof tree made of single daimon rule such that* Dai $\to_s^* \pi$.

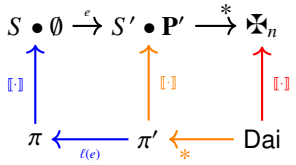
Lemma 40. *Let π' be a proof of schemes. Given a reduction $S \bullet \mathbf{P} \xrightarrow{e}_{\mathsf{P}} S' \bullet \mathbf{P}'$ such that $S' = [\![\pi']\!]^\dagger$ then there exists a proof π such that: (1) $S = [\![\pi]\!]^\dagger$ and (2) $\pi' \to_s^{\ell(e)} \pi$ where the proof search step acts on the subformula associated with the conclusion of the link e and proof search for connective of e that is $\ell(e)$.*

Equivalently the following diagram commutes (the dashed arrows represents the existence of such π):



We prove the lemma by treating all cases of reduction, this makes 4 cases for connectives and one case for the rerouting rule. For each cases, we show that the diagram commute as is illustrated in Figure 18, Figure 19 and Figure 20.

Proof of Lemma 34. Let S be a proof net, i.e. $S \to_{\mathsf{P}}^* \Uparrow_n$, reason by induction on the length n of the parsing sequence. If $n = 0$ the net S represents a proof tree of $\mathsf{MLL}_2^{\Uparrow\dagger}$ consisting of a single daimon rule; If $n > 0$ we conclude by using Lemma 40: decompose the parsing reduction as $S \bullet \emptyset \to_{\mathsf{P}} S' \bullet \mathbf{P}' \to_{\mathsf{P}}^* \Uparrow_n$ and complete the diagram:



The proof tree Dai consists of a single daimon rule and the corresponding (red) arrow is obtained as in the case $n = 0$; the proof tree π' and the corresponding (orange) arrows is obtained by applying the induction hypothesis on the top right corner, finally the proof tree π and the corresponding (blue) arrows is obtained by Lemma 40. □

Remark 41. In presence of second order quantifiers, one can substitute a cut between A and $A^\perp$ by a tensor and an existential quantifier. This is completely standard in the typed case: it adds in the sequent conclusion as many $\exists X X \otimes X^\perp$ as the number of cuts occuring in the net. In our case (which is untyped) this corresponds to substituting the links $\langle p_1, p_2 \rhd_{\mathsf{cut}}\rangle$ with the sum of links $\langle p_1, p_2 \rhd_\otimes p\rangle + \langle p \rhd_\exists q, \mathsf{s}\rangle + \langle \mathsf{s} \blacktriangleright_\exists^+ p_1\rangle + \langle \mathsf{s} \blacktriangleright_\exists^- p_2\rangle$. We can thus straightforwardly extend our sequentialisation theorem in the presence of cuts.

VII. Towards a static criterion

A. Further study of the reduction

Proposition 42. *The parsing rewriting is strongly confluent on proof nets.*

Remark 43. The fact that the parsing rewriting is strongly confluent means that if a net S is a proof net then any strategy of reduction for the parsing rewriting $\to_{\mathsf{P}}$ will terminate and normalise S to $\Uparrow_k$. As a consequence since the parsing rewriting always decrease the number of connective links or existential pointers in S, the parsing criterion can run in quadratic time, by applying a naive strategy: we look for a parsing-redex e in S (this costs a reading of S) reduce S as $S \bullet \xrightarrow{e}_{\mathsf{P}} S' \bullet \mathbf{P}'$; now do the previous step in $S' \bullet \mathbf{P}'$ (if it isn't a single daimon link). This will make n reduction steps and read the net n times where n is the number of links and existential pointers in S in S.

Given a forall link $\langle p \rhd_\forall q, \mathsf{s}\rangle$ the universal pointers $\langle \mathsf{s} \blacktriangleright_\forall t\rangle$ where t is not above s are called *out of range*. A *pointer guard* of a pointer $\langle \mathsf{s} \blacktriangleright_\forall t\rangle$ is an existential pointer $\langle \mathsf{s}' \blacktriangleright_\exists t'\rangle$ such that t' is below t; the *link* of a pointer guard is the existential link containing s'; the *guard* of a pointer $\mathbf{p} = \langle \mathsf{s} \blacktriangleright_\forall t\rangle$ is the link of its pointer guard of target t_0 such that t_0 is above the target of all pointer guards of $\mathbf{p}$ (i.e. its the exists which points below t and is the most highest in the syntax tree); we denote it $\mathsf{g}(\mathbf{p})$. we call a *potential sequence* a sequence $(\alpha_i)_{1\le i\le n}$ of links and positions. A potential sequence of a net S is a *reduction sequence* when there exists nets $(S_i)_{1\le i\le n}$ such that $S \xrightarrow{\alpha_1}_{\mathsf{P}} S_1$ and $S_i \xrightarrow{\alpha_{i+1}}_{\mathsf{P}} S_{i+1}$ for each $1 \le i \le n-1$, then the net S_n

$$\frac{}{F_1(\vec{X_1}),\ldots,F_n(\vec{X_n})}\,\text{Ⅎ} \qquad \frac{S_1,S_2,\Gamma}{S_1 \parr S_2,\Gamma}\,\parr \qquad \frac{S_1,\Gamma \quad S_2,\Delta}{S_1\otimes S_2,\Gamma,\Delta}\,\otimes \qquad \frac{S[Z/Y],\Gamma}{\forall XS[X/Y],\Gamma}\,\forall \qquad \frac{S[S'/Y],\Gamma}{\exists XS[X/Y],\Gamma}\,\forall$$

Fig. 10: Rules generating the scheme sequent calculus that we denote $\mathsf{MLL}_2^{\text{Ⅎ}\dagger}$. To apply the forall rule one must verify that all the schemes in Γ do not contain the variable X in their variable list.

$$\frac{}{\Gamma,F(X_1,\ldots,X_n)}\,\text{Ⅎ} \to_s \frac{\dfrac{}{\Gamma,F_1(X_1^1,\ldots,X_{n1}^1),F_2(X_1^2,\ldots,X_{n2}^2)}\,\text{Ⅎ}}{\Gamma,F_1(X_1^1,\ldots,X_{n1}^1)\parr F_2(X_1^2,\ldots,X_{n2}^2)}\,\parr \qquad \frac{}{\Gamma,F(X_1,\ldots,X_1),\Delta}\,\text{Ⅎ} \to_s \frac{\dfrac{}{\Gamma,F_1(X_1^1,\ldots,X_{n1}^1)}\,\text{Ⅎ} \quad \dfrac{}{\Delta,F_2(X_1^2,\ldots,X_{n2}^2)}\,\text{Ⅎ}}{\Gamma,F_1(X_1^1,\ldots,X_{n1}^1)\otimes F_2(X_1^2,\ldots,X_{n2}^2),\Delta}\,\otimes$$

$$\frac{}{\Gamma,F(X_1,\ldots,X_1)}\,\text{Ⅎ} \to_s \frac{\dfrac{}{\Gamma,F_0(X_1,\ldots,X_n,(0+1).X)}\,\text{Ⅎ}}{\Gamma,\forall XF_0(X_1,\ldots,X_n)}\,\forall \qquad \frac{}{\Gamma,F(X_1,\ldots,X_n)}\,\text{Ⅎ} \to_s \frac{\dfrac{}{\Gamma,F_0(X_1,\ldots,X_n)}\,\text{Ⅎ}}{\Gamma,\exists XF_0(X_1,\ldots,X_1)}\,\exists$$

$$\pi\left[\frac{}{\Gamma,F(X_1,\ldots,X_1)}\,\text{Ⅎ}\right] \to_s \pi[F \leftarrow_\exists W]\left[\frac{}{\Gamma,F(X_1,\ldots,X_n,Y_1,\ldots,Y_k)}\,\text{Ⅎ}\right]$$

Fig. 11: Proof search rewriting in the scheme sequent calculus. In the forall-rule the variable Y is fresh. The last rule will be related to the pointer rule of the parsing reduction. It involves a substitution $[F \leftarrow_\exists W]$ which denotes the fact of adding F as a witness to the $\exists$-quantifier of variable W thus given a F in $\mathrm{Var}^{(2)}$ it behaves as $\exists WA[F \leftarrow_\exists W] = \exists WA[F \leftarrow W]$ while passing through every other connective;d for instance $(\exists ZA)[F \leftarrow_\exists W]$ equals $\exists Z(A[F \leftarrow_\exists W])$ while $X[F \leftarrow_\exists W]$ returns X.

is the *reduct* of the sequence we denote then $S \xrightarrow{\vec{\alpha}}_{\mathsf{P}} S_n$. A reduction sequence *verifies* a net S if $S \xrightarrow{\vec{\alpha}}_{\mathsf{P}} \text{Ⅎ}_k$, then it is called a verification sequence. A reduction sequence $(\alpha_i)_{1\le i\le n}$ on S satisfies the existential precedence if for each α_i that is a forall-link the sequence $(\alpha_1,\ldots,\alpha_{i-1})$ contains the guards of each out of range pointers of α_i.

Proposition 44. *If P is a proof net, each of its verification sequence satisfies the "existential precedence rule".*

Proof. A forall-link e is contractible only when the pointers which are not above itself are in its daimons are of the form $\langle \mathsf{s} \dashrightarrow_\forall p\rangle$ i.e. they have been deactivated, however the only way pointers $\langle \mathsf{s} \dashrightarrow_\forall p\rangle$ is if they are the residuals of dependency pointers $\langle \mathsf{s} \rightsquigarrow_\forall \mathsf{s}'\rangle$ and s' is the target of an exists link e'. As a consequence it is necessary that the link e' is contracted before e. □

Definition 45. A *kingdom* of a set of positions P inside a net S is a minimal sub-proof net of S containing the set of positions P. The kingdom of a universal link e in S is a kingdom of e and all its guards (that is a kingdom of the set obtained by taking the union of the source sets and targets sets of these links).

Proposition 46. *Given a net S and $\vec{\alpha}$ is a reduction sequence for S to a daimon, then for all $\forall$ link e in $\vec{\alpha}$, the daimon link produced by $\vec{\alpha}_{\le e}$ is associated with a proof net containing a kingdom of the link e.*

The *weak parsing* is the parsing rewriting of Figure 9 but such that the forall-step can be performed as long as the forall link is below a daimon link (disregarding its pointers or ghosts). A weak reduction sequence for S is a sequence of links and pointers $(\alpha_i)_{1\le i\le n}$ such that there exists nets $(S_i)_{1\le i\le n}$ with $S = S_1$ and for all i; $S_i \xrightarrow{\alpha_i}{}^{w}_{\mathsf{P}} S_{i+1}$. A sequence $\vec{\alpha}$ is full for S if it contains all the links and existential pointers of S. Our next theorem shows how a condition on a weak reduction sequence can make it a reduction sequence; we say that a weak reduction sequence $\vec{\alpha}$ satisfies the kingdom property if for each forall-links e in $\vec{\alpha}$ we have that $\alpha_{<e}$ contains a kingdom of e and reduces it into a single daimon, the one that lies above e.

Theorem 47. *Given a guarded net S (i.e. all its out of range universal pointers have a guard) and a full weak reduction sequence $\vec{\alpha}$ for S; if $\vec{\alpha}$ satisfies the kingdom property then $\vec{\alpha}$ is a verification sequence of S (hence S is a proof net).*

B. A static correctness criterion

Based on the previous study of reduction, we define the following notion of switching for second-order multiplicative nets where the forall-link behaves similarly to a par-link.

Definition 48. The switching rewriting is defined on $\mathsf{MLL}_2^{\text{Ⅎ}}$ nets as follow:

$$S + \langle p_1, p_2 \rhd_\parr p\rangle \to_\parr S[p_1 \leftarrow p]$$
$$S + \langle p_1, p_2 \rhd_\parr p\rangle \to_\parr S[p_2 \leftarrow p]$$
$$S + \langle p \rhd_\forall q, \mathsf{s}\rangle + \textstyle\sum_{1\le i\le n}\langle \mathsf{s} \blacktriangleright_\forall t_i\rangle \to_\forall S[p \leftarrow q]$$
$$S + \langle p \rhd_\forall q, \mathsf{s}\rangle + \textstyle\sum_{1\le i\le n}\langle \mathsf{s} \blacktriangleright_\forall t_i\rangle \to_\forall S[p \leftarrow \mathsf{g}(\langle \mathsf{s} \blacktriangleright_\forall t_i\rangle)]$$

A *switching* is a normal form with respect to the switching rewriting. The switchings of a net S are all the normal forms of S with respect to the switching rewriting. The graph associated with an hypergraph $(V,E,\mathrm{s},\mathrm{t},\ell)$ is the graph whose nodes are $V \uplus E$ and such that an edge occur between a node e and p $\mathrm{s}(e)$ or $\mathrm{t}(e)$ contains p.

We conjecture that the following result holds.

Conjecture 49. *A guarded $\mathsf{MLL}_2^{\text{Ⅎ}}$ net is a proof net iff every second-order switching defines a connected and acyclic graph.*

We believe that a proof of this result could be obtained by refining the following method. First, prove that there exists at least one $\forall$ link whose kingdom does not contain another $\forall$-link (if this is not the case this will create a cycle in the switching graphs). We take this $\forall$ link. We replace it by a wide $\parr$ link connecting the corresponding $\exists$ links. We check that this change does not impact the correction graphs (same switchings). This shows that there is a subproof (multiplicative criterion) ending with this $\parr$. To continue, we remove the $\forall$ link we just treated (keep it as a wide $\parr$), and continue with next $\forall$ link without other $\forall$ link above.

This would establish that if the criterion is satisfied, it is a proof net. Conversely, showing that any proof net satisfies the criterion should be straightfoward.

References

[1] J.-Y. Girard, "Linear logic," *Theoretical Computer Science*, vol. 50, no. 1, pp. 1 – 101, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0304397587900454

[2] ——, "Normal functors, power series and λ-calculus," *Annals of Pure and Applied Logic*, vol. 37, no. 2, pp. 129–177, 1988. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0168007288900255

[3] R. Banach, "Sequent reconstruction in llm—a sweepline proof," *Annals of Pure and Applied Logic*, vol. 73, no. 3, pp. 277–295, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/016800729400033Y

[4] D. Bechet, "Minimality of the correctness criterion for multiplicative proof nets," *Mathematical Structures in Computer Science*, vol. 8, no. 6, p. 543–558, 1998.

[5] P. J. d. Naurois and V. Mogbil, "Correctness of multiplicative additive proof structures is nl-complete," in *2008 23rd Annual IEEE Symposium on Logic in Computer Science*, 2008, pp. 476–485.

[6] V. Danos, "La logique linéaire appliquée à l'étude de divers processus de normalisation (principalement du lambda-calcul)," Ph.D. dissertation, Université Paris 7, 1990, thèse de doctorat dirigée par Girard, Jean-Yves Mathématiques Paris 7 1990. [Online]. Available: http://www.theses.fr/1990PA077188

[7] V. Danos and L. Regnier, "The structure of multiplicatives," *Archive for Mathematical Logic*, vol. 28, no. 3, pp. 181–203, 10 1989. [Online]. Available: https://doi.org/10.1007/BF01622878

[8] C. Retoré, "Perfect matchings and series-parallel graphs: multiplicatives proof nets as r& b-graphs: [extended abstract]," *Electronic Notes in Theoretical Computer Science*, vol. 3, pp. 167 – 182, 1996, linear Logic 96 Tokyo Meeting. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066105804165

[9] S. Guerrini and A. Masini, "Parsing mell proof nets," *IRCS Technical Reports Series*, vol. 254, 03 2001.

[10] J.-Y. Girard, "Quantifiers in linear logic ii," *Nuovi problemi della logica e della filosofia della scienza*, vol. 2, p. 1, 1991.

[11] L. STRAßBURGER, "Deep inference and expansion trees for second-order multiplicative linear logic," *Mathematical Structures in Computer Science*, vol. 29, no. 8, p. 1030–1060, 2019.

[12] A. Ragot, T. Seiller, and L. Tortora de Falco, "Linear realisability over nets: multiplicatives," *(To appear in) 33rd EACSL Annual Conference on Computer Science Logic (CSL 2025)*., 2024.

[13] J.-Y. Girard, "Linear logic," *Theoretical computer science*, vol. 50, no. 1, pp. 1–101, 1987.

[14] S. Guerrini, "A linear algorithm for mll proof net correctness and sequentialization," *Theoretical Computer Science*, vol. 412, no. 20, pp. 1958–1978, 2011, girard's Festschrift. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397510007127

[15] J.-Y. Girard, "Locus solum: From the rules of logic to the logic of rules," in *Computer Science Logic*, L. Fribourg, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 38–38.

[16] P. Curien, "Introduction to linear logic and ludics, part II," *CoRR*, vol. abs/cs/0501039, 2005. [Online]. Available: http://arxiv.org/abs/cs/0501039

[17] A. Guglielmi, "A system of interaction and structure," *ACM Transactions on Computational Logic*, vol. 8, no. 1, p. 1, Jan. 2007. [Online]. Available: http://dx.doi.org/10.1145/1182613.1182614

[18] L. Straßburger, "Some observations on the proof theory of second order propositional multiplicative linear logic," in *Typed Lambda Calculi and Applications*, P.-L. Curien, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 309–324.

[19] A. Doumane, "On the infinitary proof theory of logics with fixed points," Theses, Université Sorbonne Paris Cité, Jun. 2017. [Online]. Available: https://hal.science/tel-01676953

Contents

Appendix

A. Related Works

We discuss a related line of work proposed by L. Strassburger which stems from deep inference. A closely related line of work, based on the formalism of deep inference [17] has been proposed by L. Strassburger, initiated in [18] and continued in [11], where the author defines proof structures (called *pre-proof graphs*) for second order multiplicative linear logic with units and atomic axioms, and proposes a correctness criterion characterising the pre-proof graphs which represent a proof from the sequent calculus.

Strassburger's approach and ours provide complementary perspectives that mutually enrich and clarify one another. While his work shares similarities with ours, key differences set our contributions apart, which we outline below:

a) Witnesses of existential quantifiers: The pre-proof graphs defined in [11] are inherently typed, and make uses of atomic axioms, as a consequence, the witnesses[1] of existential quantifiers have the same syntax tree. By contrast, our approach only requires the witnesses of existential quantifiers to be coherent (see Figure 4). Furthermore the witnesses of pre-proof graphs (as specified in [11] Definition 4.3 condition 2) cannot capture bounded variables: indeed this is the case of proof structures representing a proof, however this condition outright rejects some kind of proof structures that can be defined in our setting. For instance the two left-most proof structures of Figure 7 cannot be defined in Strassburger's framework but are permitted in ours, though ultimately rejected by our correctness criterion. A consequence of these two first observations is that we define a greater set of proof structures, namely there are more proof structures than pre-proof graphs. This, in particular, has consequences when investigating cut-elimination, and more specifically, when one wants to characterise the behavior of a proof structure using cut-elimination: because some behaviors can only be revealed when using the cut elimination with incorrect proof structures (see [12]).

b) Time complexity of the correctness criteria: Strassburger's correctness criterion for pre-proof graphs is based on the Danos-Regnier criterion [7]: its naive implementation is an algorithm running in exponential time. In contrast, our parsing-based criterion terminates in quadratic time (Remark 43).

c) Proof Equivalence: The criterion for pre-proof graphs reconstructs a deep inference proof from a graph, because a deep inference proof may correspond to many sequent calculus proof the proof-equivalence induced by pre-proof-graphs is unclear (as observed in [11] Section 8). However, a version based on sequent calculus is provided in the report version of [11], but the equivalence induced on sequent calculus is still unclear because of the use of (equivalents of) $\forall$-boxes, therefore some commutations (e.g. ($\forall/⅋$) and ($\forall/\forall$)) are not directly captured by the pre-proof graph syntax and must involve a rewriting of the boxes. On the other hand our criterion reconstructs a proof from the sequent calculus without relying on boxes. Thus our approach induces natural proof equivalences avoiding the complications associated with box rewriting (Remark 19).

d) The presence of daimons: As pointed out in the subsection II-C we consider proof structures in the presence of daimons and our criterion characterise $\mathsf{MLL}_2^{⊹}$ proof-nets,

[1] To be more precise those with the same polarity.

whereas generalised axioms are absent in pre-proof graphs of [11].

B. Complements to Section II

1) Occurences of a formula: The aim of this additional subsection is to define the occurences of MLL_2 formulas within another formula, a sequent, or a proof tree (from $\mathsf{MLL}_2^{\Uparrow}$ or MLL_2). We introduce three symbols called *directions* to describe paths within syntax trees, l, r and up which respectively correspond to the following instructions "go left", "go right" and "go up". A sequence ρ in $\{\mathrm{l}, \mathrm{r}, \mathrm{up}\}^*$ is an *address* or a *path*. The occurence of a formula A in a formula B is given by an *address*: more precisely, directions act on formulas in the following way (where $\square$ is a binary connective i.e. $\otimes$ or $\parr$ and Q is a quantifier i.e. $\exists$ or $\forall$):

$$\begin{array}{ll} \mathsf{d} \cdot X & \triangleq \text{undefined} \\ \mathrm{l} \cdot A \square B & \triangleq A \\ \mathrm{r} \cdot A \square B & \triangleq A \\ \mathrm{up} \cdot A \square B & \triangleq \text{undefined} \end{array} \qquad \begin{array}{ll} \mathrm{l} \cdot QX\,A & \triangleq \text{undefined} \\ \mathrm{r} \cdot QX\,A & \triangleq \text{undefined} \\ \mathrm{up} \cdot QX\,A & \triangleq A \end{array}$$

Above A and B are MLL_2 formulas and X is a propositional variable.

To identify an occurrence of a formula in a sequent Γ, we use a pair (i, ρ) made of an *index* i to identify the position of the formula in $\Gamma = A_1, \ldots, A_n$ and a *path* $\rho \in \{\mathrm{l}, \mathrm{r}, \mathrm{up}\}^*$ to locate the formula within the syntax tree of A_i. For instance, in $\vdash \Gamma = \vdash P \otimes Q, \forall X R, P$, $(0, \mathrm{l})$ and $(2, \varepsilon)$ are the two occurences of P while $(1, \mathrm{up})$ is the only occurence of R. Similarly, to identify an *occurence of a formula* within a proof tree π of MLL_2 we use a tuple (ξ, i, ρ) made of a *path* $\xi \in \{\mathrm{l}, \mathrm{r}, \mathrm{up}\}^*$ which locates a sequent Γ within π together with an occurence (i, ρ) locating a formula within Γ.

We have presented the notion of occurence for formulas and proof trees of MLL_2 and $\mathsf{MLL}_2^{\Uparrow}$, but indeed this can be adapted for $\mathsf{MLL}_2^{\Uparrow\dagger}$ and *schemes* of section VI rather than formulas.

2) Threads: Another technical aspect of sequent calculus is that of threads, that we briefly expose, because the thread of a formula A in a proof π will corresponds to a position in the hypergraph $[\![\pi]\!]$.

To define threads one must observe that the occurrences of a formula a same formula A within a proof tree π can be tracked using the trace function tr. The *trace function* is a total function that maps occurrences of formulas in the premisses of a rule to the corresponding occurrences in the conclusion of that rule (we do not illustrate it here but this is standard especially in the works involving fixed points, for instance see [19]). A *thread* in π is a sequence $(x_1, \ldots, x_n)$ of occurences of the same formula such that $\mathsf{tr}(x_i) = x_{i+1}$ for $1 \leq i \leq n-1$; A thread is maximal if it cannot be extended, i.e., it is not the prefix or suffix of any other thread in π. Every occurence α of a formula in a proof tree π has a unique associated maximal thread that we denote $\vec{\pi}(\alpha)$.

Definition 50 (Active and accessible occurences)**.** The occurence α of a formula in a proof tree is *active* if (1) a daimon rule introducing $A_1, \ldots, A_n$ occurs in π and α is the occurence of one the formulas A_i or (2) if there exists a rule in π such that the occurence α is the main formula of the rule.

An occurence α of a formula in a proof tree is *accessible* if one of the occurences in $\vec{\pi}(\alpha)$ is *active*.

Occurrences and threads can be cumbersome to formalize rigorously. An alternative approach is to use a *localised version of the sequent calculus*, where sequents are composed of syntax trees rather than formulas. In this setting, an occurrence of a formula within the proof tree π corresponds directly to a node within a syntax tree. This eliminates the need for explicit paths or indices, as the structure of the syntax tree inherently identifies the location of the formula.

3) Eta expansion: The eta expansion is defined on MLL_2 proofs as the rewriting in Figure 12. Note that to obtain a complete proof the eta expansion of quantifiers will never be needed, because an exists cannot have for witness a (strict) subformula of $\forall XA$ or of $\exists XA$.

C. Complements to Section III

1) Complements on coherence: We here provide a more detailed formalisation of the notion of coherence and anticoherence of position, by defining the coherence of subtrees so that, two positions are coherent if and only if the subtrees above the two positions are coherent. For this, we introduce the notion of *tree with pointers*. This also illustrate that when testing the coherence of the trees above a position p we may need to hide some of the subtrees above p as illustrated in Figure 13.

Definition 51. Given a regular position p, *trees with pointers* of conclusion p are inductively defined as follows:

- If $S = \langle p \rangle$ is a trivial hypergraph made of a single position p, and $\mathbf{P}(p)$ is a hypergraph consisting only of universal pointers having p as target, then $\langle p \rangle + \mathbf{P}(p)$ is a tree with pointer of conclusion p.
- If $\mathcal{S}$ is a tree with pointers of conclusion p, $\mathcal{S}'$ is a tree with pointers of conclusion p' disjoint from $\mathcal{S}$, and q is a fresh position: $\mathcal{S} \cup \mathcal{S}' + \langle p, p' \rhd_{\otimes} q \rangle$ and $\mathcal{S} \cup \mathcal{S}' + \langle p, p' \rhd_{\parr} q \rangle$ are trees with pointers of conclusion q;
- If S is a tree with pointers of conclusion p, q is a fresh position, and s is a pointer position (which could occur in pointers of S), then $\mathcal{S} + \langle p \rhd_{\forall} q, \mathsf{s} \rangle$ is a tree with pointers of conclusion q.

Given a pre-net S and a position p, we now define in a straightforward way *the tree (with pointers) above p in* $(S, \mathbf{P})$, that we denote $\mathcal{T}_p(N)$. This will then be used to define the syntactic tree above the vertex, including the pointers of $\forall$ links that may point to vertices in that syntactic tree. We seek to ensure that two existential pointers are coherent of respective target p and q: for that *it is not enough* to look at the trees with pointers above p and q because the existential links above p and q may instantiate subtrees which have no more influence on the types (Figure 13), this is why we will introduce the notion of *type trees* above p.

$$\dfrac{}{A \parr B, A^\perp \otimes B^\perp}\,\text{ax} \to_\eta \dfrac{\dfrac{}{A, A^\perp}\,\text{ax} \quad \dfrac{}{B, B^\perp}\,\text{ax}}{\dfrac{A, B, A^\perp \otimes B^\perp}{A \parr B, A^\perp \otimes B^\perp}\,\parr}\,\otimes$$

$$\dfrac{}{\forall XX, \exists YY^\perp}\,\text{ax} \to_\eta \dfrac{\dfrac{}{Z, Z^\perp}\,\text{ax}}{\dfrac{Z, \exists YY^\perp}{\forall XX, \exists YY^\perp}\,\exists}\,\forall$$

Fig. 12: Eta expansion in MLL_2.

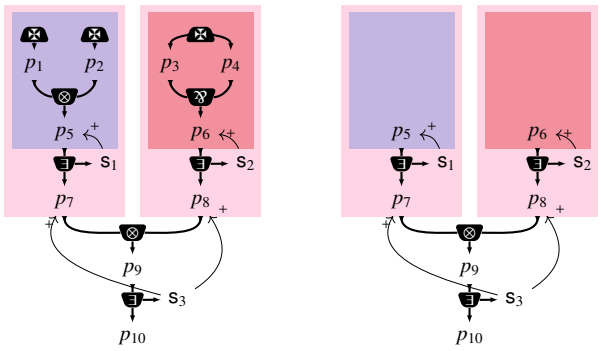


Fig. 13: The coherence of witnesses checks the coherence of partial trees: to verify that the last existential link of the left-hand side pre–net has coherent witnesses one verifies that the last exisential link in the right hand side hypergraph has coherent witnesses, i.e. that the trees above the targets of existential pointers without hidden trees are coherent.

Definition 52. Given a pre-net S and a position p, *the tree (with pointers) above p in S*, written $\mathcal{T}_p(S)$, is defined as the largest p–tree $\mathcal{S}$ that is contained in $S + \mathbf{P}$ when its daimon links are removed. A *hidden tree* T_0 in a tree with pointers T is a tree with pointers above a position t such there exists an existential pointer $\langle \mathsf{s} \blacktriangleright_\exists t\rangle \in \mathbf{P}$ that is fully defined on T. The *type tree* of p in $(S, \mathbf{P})$ is the tree denoted $\mathsf{T}_p(N)$ obtained by taking $\mathcal{T}_p(N)$ and removing each of its hidden trees. Given an existential link $\langle p \rhd_\exists q, \mathsf{s}\rangle$ its *positive (resp. negative) type trees* are the trees of the set $W_S^+(\mathsf{s}) = \{\mathsf{T}_r(N) \mid \langle \mathsf{s} \blacktriangleright_\exists^+ r\rangle\}$ (resp. $W_S^-(\mathsf{s})\{\mathsf{T}_r(N) \mid \langle \mathsf{s} \blacktriangleright_\exists^- r\rangle\}$).

Remark 53. In a type tree universal and existential pointers targets are regular positions which are the target of no non–pointer links.

Definition 54. Given a net S we define the tree with pointer above a regular position p of S inductively

We must now defined the coherence of subtrees. To do so we first define when two subtrees are *unifiable*. Given a tree with pointers $\mathcal{T}$ we can easily associate it a formula of MLL_2 called position-formula[2], denoted $\mathrm{F}(\mathcal{T})$ whose propositional variables are pairs $p(\vec{X})$ made of a position together with a finite sequence of port position, one does so by induction:

[2]In fact these really are similar to the schemes of subsection VI-A

$$\begin{array}{lcl}
\mathrm{F}(\langle p\rangle + \mathbf{P}) & \triangleq & p(\mathrm{s}(\mathbf{P})) \\
\mathrm{F}(\mathcal{T}_1 + \mathcal{T}_2 + \langle p_1, p_2 \rhd_\otimes p\rangle) & \triangleq & \mathrm{F}(\mathcal{T}_1) \otimes \mathrm{F}(\mathcal{T}_2) \\
\mathrm{F}(\mathcal{T}_1 + \mathcal{T}_2 + \langle p_1, p_2 \rhd_\parr p\rangle) & \triangleq & \mathrm{F}(\mathcal{T}_1) \parr \mathrm{F}(\mathcal{T}_2) \\
\mathrm{F}(\mathcal{T}_1 + \langle p \rhd_\forall q, \mathsf{s}\rangle) & \triangleq & \forall \mathsf{s}\mathrm{F}(\mathcal{T}_1) \\
\mathrm{F}(\mathcal{T}_1 + \langle p \rhd_\exists q, \mathsf{s}\rangle + \mathbf{P}) & \triangleq & \exists \mathsf{s}\mathrm{F}(\mathcal{T}_1)
\end{array}$$

Two trees $\mathcal{T}_1$ and $\mathcal{T}_2$ are *unifiable* if their formulas $\mathrm{F}(\mathcal{T}_1)$ and $\mathrm{F}(\mathcal{T}_2)$ are unifiable, i.e. there exists a substitution θ such that $\theta\mathrm{F}(\mathcal{T}_1)$ and $\theta\mathrm{F}(\mathcal{T}_2)$ are equal. Note such substitutions θ are compositions of subsitutions of the form $[p(\vec{X}) \leftarrow \mathrm{F}(\mathcal{T})]$. Because unifiability does not take into account the information of the pointers it is not sufficient to define coherence. Given a position-formula we define its *occuring pointers* as follow:

$$\begin{array}{lcl}
\mathrm{p}_\forall(p(\vec{X})) & \triangleq & \{x \in \vec{X}\} \\
\mathrm{p}_\forall(F \square G) & \triangleq & \mathrm{p}_\forall(F) \cup \mathrm{p}_\forall(G) \\
\mathrm{p}_\forall(\forall \mathsf{s} F) & \triangleq & \mathrm{p}_\forall(F) \setminus \mathsf{s} \\
\mathrm{p}_\forall(\exists \mathsf{s} F) & \triangleq & \mathrm{p}_\forall(F)
\end{array}$$

A simple substitution $[p \leftarrow F]$ is *coherent* if $\mathrm{p}_\forall(p)$ equals $\mathrm{p}_\forall(F)$. A substitution is *coherent* when it is the composition of simple coherent substitutions. Finally one can defined one coherence: two trees $\mathcal{T}_1$ and $\mathcal{T}_2$ are *coherent* if the formulas $\mathrm{F}(\mathcal{T}_1)$ and $\mathrm{F}(\mathcal{T}_2)$ can be unified by a coherent substitution. Indeed the definition of coherent subtrees naturally generalises to finites sets of trees: the trees $\mathcal{T}_1, \ldots, \mathcal{T}_n$ are coherent if they can be unified by a coherent substitution.

Formula-tree can also be negated in the obvious way, which allows us to define dual-coherence.

$$\begin{array}{lcllcl}
p(\vec{X})^\perp & \triangleq & p(\vec{X}) & (\forall \mathsf{s} F)^\perp & \triangleq & \exists \mathsf{s} F^\perp \\
(F \square G)^\perp & \triangleq & F^\perp \square G^\perp & (\exists \mathsf{s} F)^\perp & \triangleq & \forall \mathsf{s} F^\perp
\end{array}$$

Note that $\mathrm{p}_\forall(F)$ and $\mathrm{p}_\forall(F^\perp)$ are equal. Two trees $\mathcal{T}_1$ and $\mathcal{T}_2$ are *dually coherent* whenever $\mathcal{T}_1$ and $\mathcal{T}_2{}^\perp$ are coherent.

Notation 55. Given a formula A and B and Ξ a set of addresses and ξ an address. We denote $\mathsf{adr}_B(A)$ the set of addresses of the subformula B in A. We denote $\mathsf{Get}(A, \xi)$ the formula at address ξ (if defined), we denote $\mathsf{Get}(A, \Xi)$ the set of formulas $\{\mathsf{Get}(A, \xi) \mid \xi \in \Xi\}$. Similarly given a net S and one of its position p we defined $\mathsf{Get}(S, p, \xi)$ and $\mathsf{Get}(S, p, \Xi)$

2) Proof of proposition 17:

Proposition 17. *For each proof π of $\mathsf{MLL}_2^{\Cross}$, $[\![\pi]\!]$ is a net.*

Proof. This is done by induction on the proof π:

- if π is a daimon rule introducing a sequent $\Gamma = A_1, \ldots, A_n$ the $[\![\pi]\!]$ is a daimon link summed with universal pointers, indeed this is a $\mathsf{MLL}_2^{\Cross}$ net.
- if π is a tensor rule applied on two proofs π_1 and π_2, we consider disjoint renamings for the hypergraphs $[\![\pi_1]\!]$

and $[\![\pi_2]\!]$, then $[\![\pi]\!] = [\![\pi_1]\!] + [\![\pi_2]\!] + \langle p, q \rhd_\otimes r\rangle$ for a fresh position r where p and q are the positions corresponding to the formulas A and B, indeed the sum of $\mathsf{MLL}_2^{\Game}$ net is still an $\mathsf{MLL}_2^{\Game}$ net and furthermore adding a terminal link which has for conclusion a fresh position results in a $\mathsf{MLL}_2^{\Game}$ net.

- if π is a parr rule applied on a proofs π_1 and p, q are the conclusions corresponding to the principal formulas in the rule, we define $[\![\pi]\!] = [\![\pi_1]\!] + \langle p, q \rhd_{\parr} r\rangle$ for a fresh position r, adding a terminal link which has for conclusion a fresh position to a $\mathsf{MLL}_2^{\Game}$ net results in a $\mathsf{MLL}_2^{\Game}$ net.
- if π is a existential quantifier rule applied on a proof π_1:

$$\dfrac{\begin{matrix}[\![\pi_1]\!] \\ \Gamma, A[B/X]\end{matrix}}{\Gamma, \exists Y A[Y/X]}\,\exists$$

and p is the conclusion associated with to the principal formula in the rule; $q_1, \ldots, q_n$ are the positions corresponding to the positive witnesses that we obtain by taking the addresses of X in A i.e. $\mathsf{adr}_X(A)$ and finding the corresponding position in the net $[\![\pi_1]\!]$; this means that $\{q_1, \ldots, q_n\} = \mathsf{Get}([\![\pi]\!], p, \mathsf{adr}_X(A))$. Similarly $r_1, \ldots, r_m$ are the positions associated with the negative witnesses $\mathsf{Get}([\![\pi]\!], p, \mathsf{adr}_{X^\perp}(A))$.
With this sets of positions defined we can define the desequentialisation of π as follow for fresh position w and pointer position s:

$$[\![\pi]\!] = [\![\pi_1]\!] + \langle p \rhd_\exists q, \mathsf{s}\rangle + \sum_{i=1}^{n} \langle s \blacktriangleright_\exists^+ q_i\rangle + \sum_{i=1}^{m} \langle s \blacktriangleright_\exists^- r_i\rangle$$

- if π is a universal quantifier rule applied on a proofs π_1 and p is the conclusion corresponding to the principal formula in the rule, and $q_1, \ldots, q_n$ are positions corresponding to the variable X (recall that we assume the existence of an injective map $X \mapsto \mathsf{s}_X$ mapping all propositional variables to a port position, thus the q_i's are the positions in $[\![\pi_1]\!]$ which are the target of a pointer of source s_X i.e. some pointer $\langle s \blacktriangleright_\forall q_i\rangle$ occur in the net). Having defined $q_1, \ldots, q_n$ we define the desequentialisation of π as $[\![\pi]\!] = [\![\pi_1]\!] + \sum_{i=1}^{n} \langle s \blacktriangleright_\forall r_i\rangle + \langle p \rhd_\forall r, s\rangle$ for fresh position r and pointer position s.

(1) By construction $[\![\pi]\!]$ is a pre–net one can inductively check that the property of target and source surjectivity are preserved because the new position is always taken fresh, furthermore target surjectivity is always preserved because the new position is always the target of the added link. (2) Further $[\![\pi]\!]$ is a net because its existsential pointers verify coherence, this is proved inductively on π, but the only interesting case is when adding an existential link and its pointers; assume the proof is of the following form

$$\dfrac{\begin{matrix}[\![\pi_1]\!] \\ \Gamma, A[B/X]\end{matrix}}{\Gamma, \exists Y A[Y/X]}\,\exists$$

An exists link $\langle p \rhd_\exists q, \mathsf{s}\rangle$ is added to $[\![\pi_1]\!]$ its conclusion is the fresh position q and it points to $\mathsf{Get}([\![\pi]\!], p, \mathsf{adr}_X(A))$ thus it points above p, furthermore the witnesses are coherent (to do so one observe that occurences of formulas B in π are mapped to a position q whose tree may be typed by B, more specifically one shows that two occurences of a same formula B in a proof π are mapped to positions in $[\![\pi]\!]$ which are coherent – on the other hand, if the formula are B and $B^\perp$ they are mapped to anti-coherent positions.). (3) We must ensure that $[\![\pi]\!]$ contains only $\forall$-pointers whose target are outputs of daimon links; this again is proved by induction straightforwardly. □

D. Proof Equivalence – On remark 19

The fact that our nets are box-free allows us to capture exactly the proof equivalence of MLL_2; if two proofs of MLL_2 are equivalent (modulo rule commutations, see Figure 14) they are represented by the same net. Since we our nets are untyped objects this becomes an equivalence modulo alpha conversion: two proof of MLL_2 are equivalent (modulo alpha conversion) if and only if they are represented by the same net.

Definition 56. Given two symbols a, b belonging to $\{\otimes, \parr, \forall, \exists\}$ the commutation $\sim_{(\mathrm{a/b})}$ is the (symmetric) binary relation on proof trees of $\mathsf{MLL}_2^{\Game}$ or MLL_2 as defined in Figure 14. Two proof trees π and π' are *1-equivalent*, denoted $\pi \sim_1 \pi'$, if there exists two symbols a, b of $\{\otimes, \parr, \forall, \exists\}$ two subtrees π_0 of π and π'_0 of π' and a proof tree ρ such that $\rho[\pi_0] = \pi$, $\pi' = \rho[\pi'_0]$ and $\pi_0 \sim_{(\mathrm{a/b})} \pi'_0$. Two proof trees are equivalent if $\pi \sim_1^* \pi'$ where $\sim_1^*$ is the transitive, symmetric and reflexive closure of $\sim_1$.

Proposition 57. *Let π and π' be two proof trees of $\mathsf{MLL}_2^{\Game}$ and V a set of propositional variables; if $\pi \sim \pi'$ then $\langle \pi; V\rangle = \langle \pi'; V\rangle$*

Proof. This is given by Figure 15 and Figure 16. □

Proposition 58. *Let π and π' be two proof trees of the scheme calculus, if $[\![\pi]\!] = [\![\pi']\!]$ then π and π' are equivalent.*

Proof. Assume that $[\![\pi]\!]$ and $[\![\pi']\!]$ are both equal to a net S. By soundness, S is a proof net i.e. $S \to \Game_n$ and we have the diagram as in Theorem 28, to conclude we show that given two equivalent proofs π and π' represented by S and two contractible link e_1, e_2 in S the proofs obtained by the steps $S \xrightarrow{e_1}_{\mathsf{P}} \xrightarrow{e_2}_{\mathsf{P}} S'$ and $S \xrightarrow{e_2}_{\mathsf{P}} \xrightarrow{e_1}_{\mathsf{P}} S'$ (the nets are equal by confluence) are equivalent proofs. □

E. Complements to section VI

1) Desequentialisation for scheme calculus: We define the desequentialisation of proof trees from the scheme calculus as in Figure 17.

2) Completeness Diagrams: We illustrate the technical lemma (Lemma 40) which leads us to prove the completeness theorem (Theorem 28) in the figures 18, 19 and 20.

Also note that the substitution and desequentialisation are compatible:

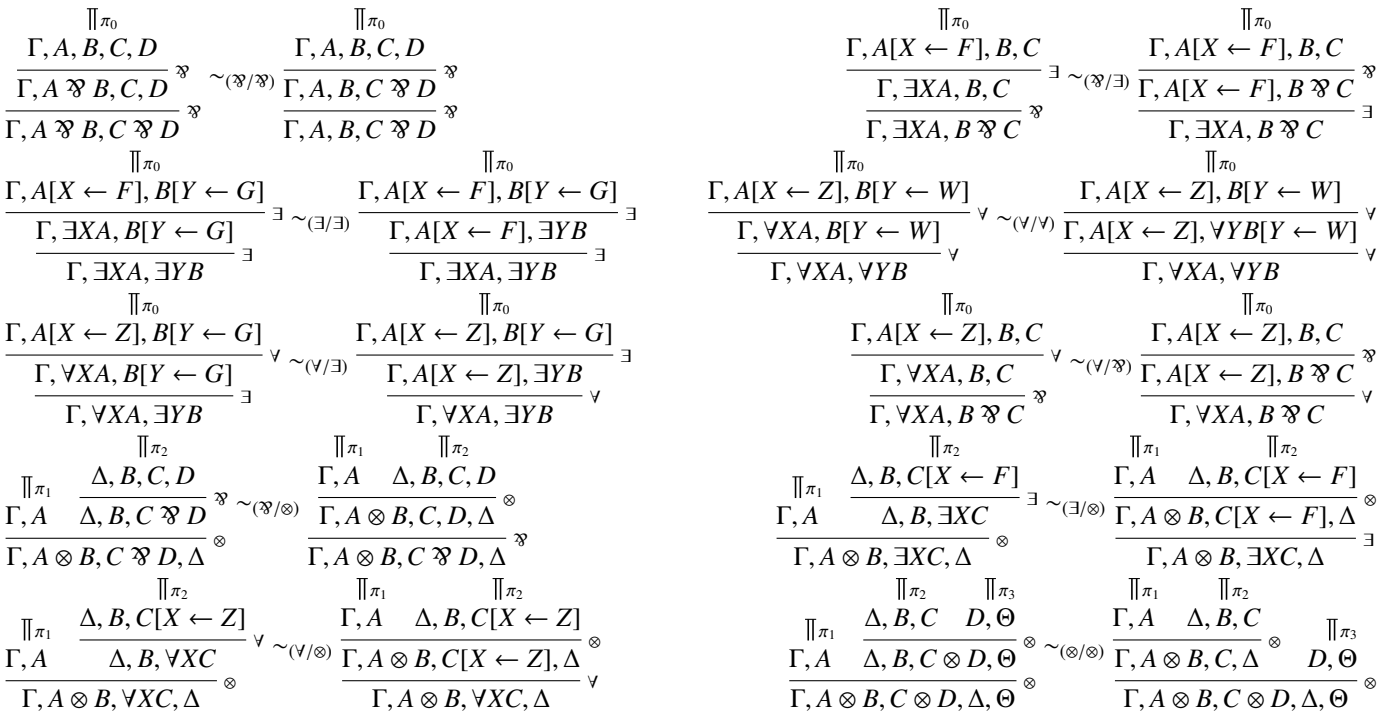


Fig. 14: The proof equivalence on MLL_2^{Π} and MLL_2 proofs. The $\forall$ rules are always supposed to be applyable meaning the variable does not appear in the context. In the commutation $(\forall/\otimes)$ we assume that the variable Y of the universal quantifier does not occur in Γ, A; in other words a tensor rule between two proofs should only be allowed when the two proofs don't variables of universal quntifiers. In the commutation $(\forall/\exists)$ the variable Z of the universal quantifier does not occur in the witness G of the existential quantifier.

Lemma 59. *Given a proof π $[\![\pi[\pi'/d]]\!] = [\![\pi]\!][[\![\pi']\!]/d]$.*

3) On remark 41: Figure 21 illustrates the substitution described in Remark 41.

F. Complements to Section VII-A

1) On confluence: Proving confluence is done by decomposing the parsing reduction as $\to_{\mathtt{ptr}} \uplus \to_{\mathsf{P}}^{C}$ where $\to_{\mathtt{ptr}}$ is the pointer rule and $\to_{\mathsf{P}}^{C}$ is the parsing rewriting without the pointer rule. One observe that $\to_{\mathsf{P}}^{C}$ contains a unique critical pair that can only occur in incorrect nets, showing that $\to_{\mathsf{P}}^{C}$ is strongly confluent on proof nets. Then one shows that $\to_{\mathtt{ptr}}$ is confluent and finally that one step of $\to_{\mathtt{ptr}}$ and one step of $\to_{\mathsf{P}}^{C}$ always commute when then can be both performed.

We present the proof of the confluence of the parsing rewriting.

Proposition 42. *The parsing rewriting is strongly confluent on proof nets.*

Proof. If both rewriting steps are contracting a connective link confluence hold by Proposition 60; If both rewriting steps are pointers rule confluence hold by Proposition 61; If one step is pointer rule while the other is a connective confluence hold by Proposition 62. □

To do so we prove a series of proposition as sketched in section VII-A. Let us decompose the parsing reduction as $\to_{\mathtt{ptr}} \uplus \to_{\mathsf{P}}^{C}$ where $\to_{\mathtt{ptr}}$ is the pointer rule and $\to_{\mathsf{P}}^{C}$ is the parsing rewriting without the pointer rule.

Proposition 60. *The parsing reduction of connective links $\to_{\mathsf{P}}^{C}$ is confluent on proof nets.*

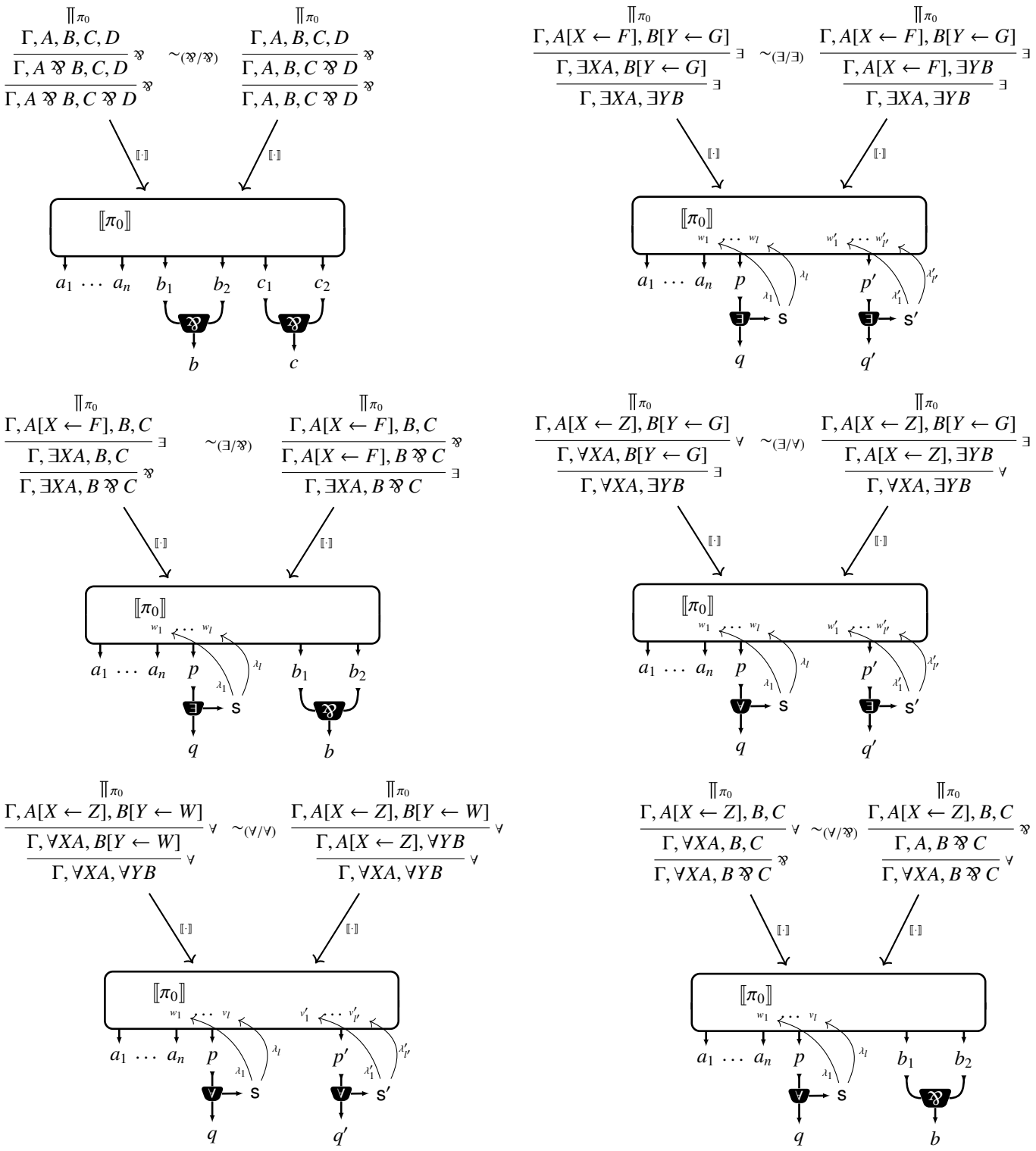


Fig. 15: Equivalent proofs have the same representation as a net, this includes all the cases which do not contain the tensor.

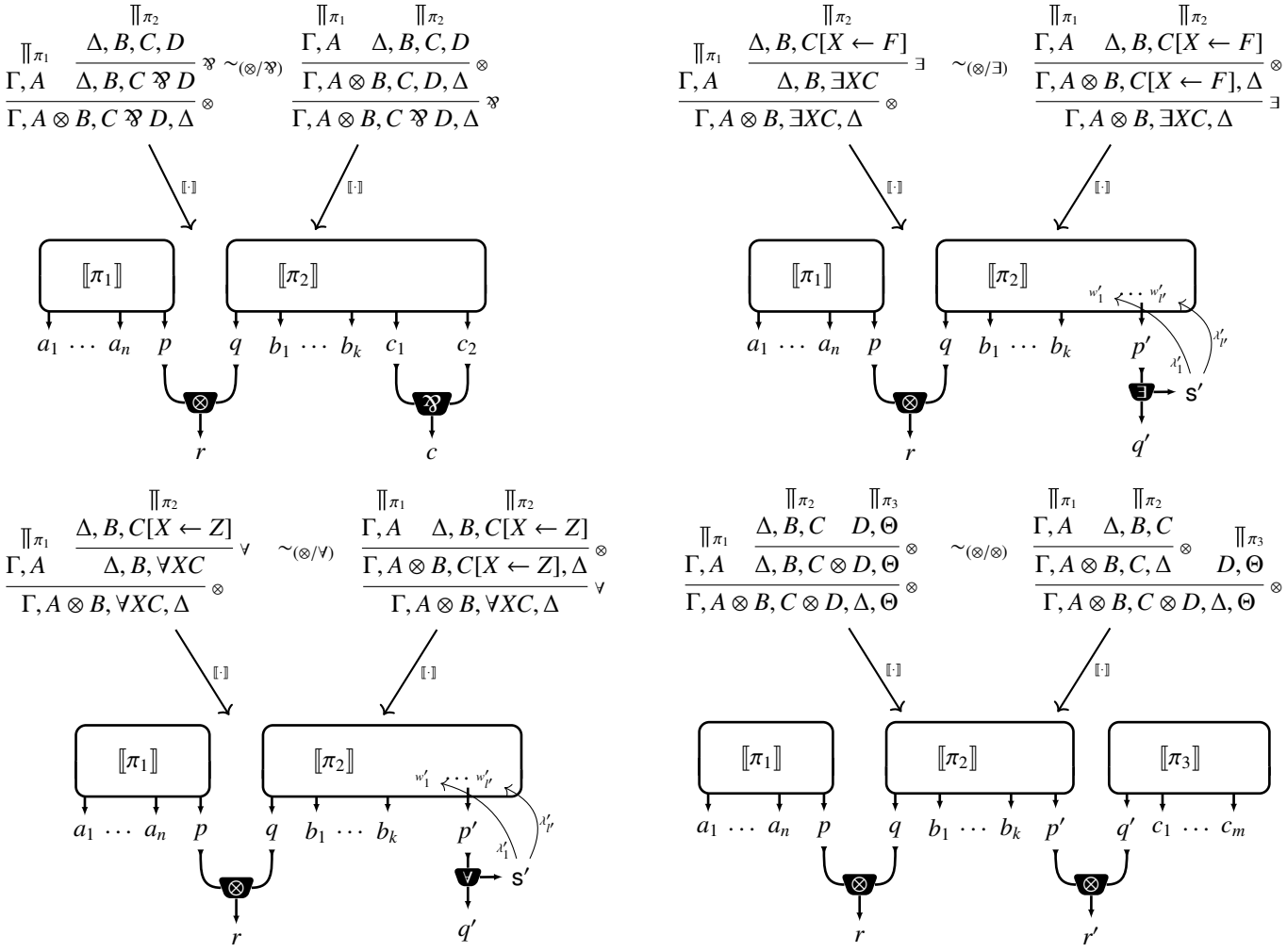


Fig. 16: Equivalent proofs have the same representation as a net: all tensor cases.

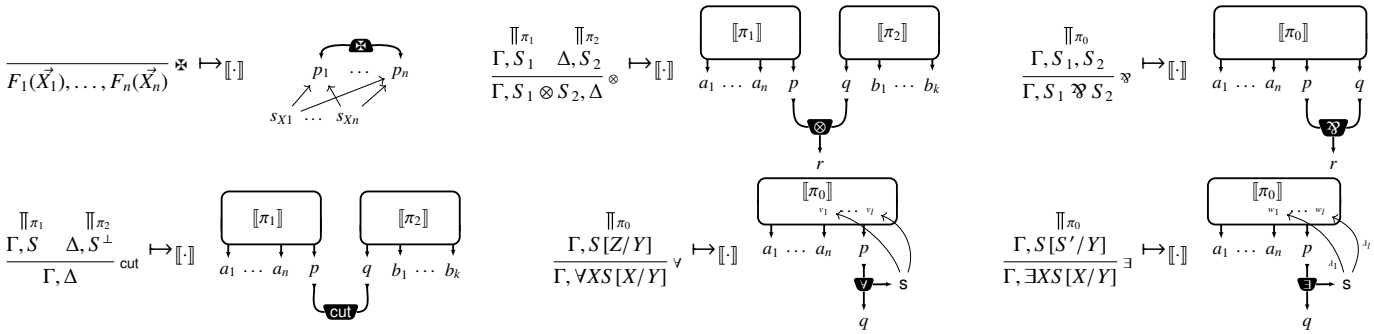


Fig. 17: The desequentialisation process of a proof π from the scheme calculus $\mathsf{MLL}_2^{\Fax\dagger}$, it associates a proof from the sequent calculus $\mathsf{MLL}_2^{\Fax\dagger}$ with a $\mathsf{MLL}_2^{\Fax}$ net. The only difference with the desequentialisation of sequent calculus lies in the base case, namely a point $\langle \mathsf{s}_x \blacktriangleright_\forall t \rangle$ is added in the base case only when t is the position associated with $F(\vec{X})$ and $\vec{X}$ contains the variable X.

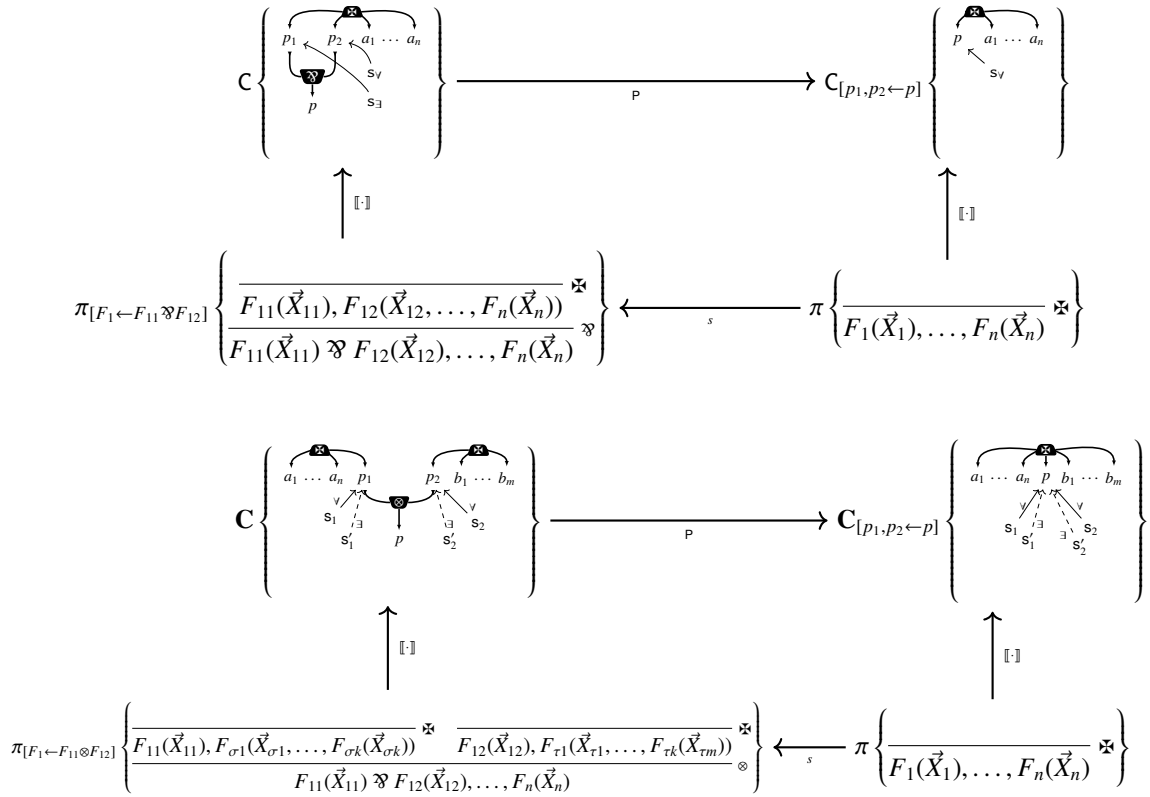


Fig. 18: Completeness Diagrams (1/3).

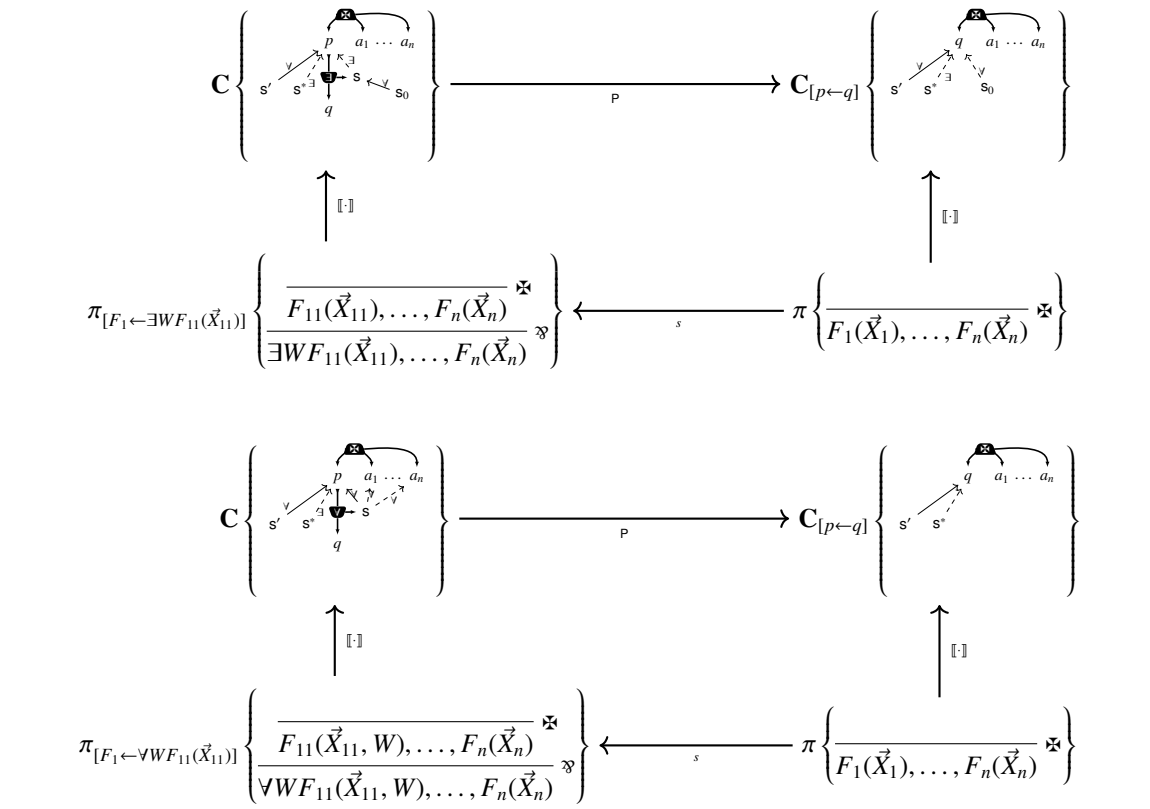


Fig. 19: Completeness Diagrams (2/3).

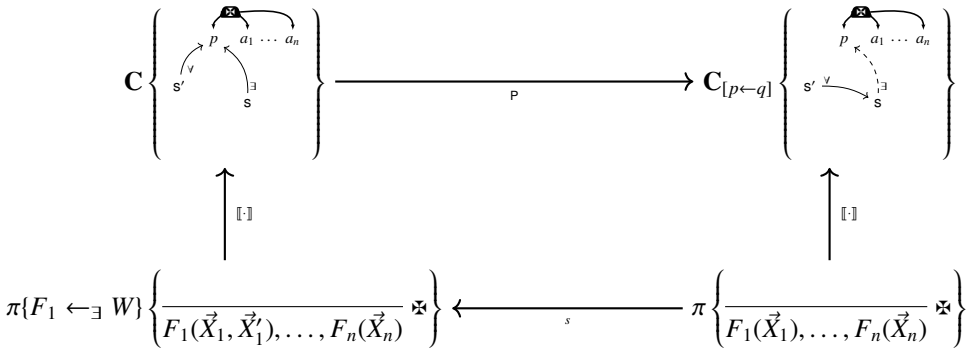


Fig. 20: Completeness Diagrams (3/3). The parsing read backwards correspond takes a formula F_1 occuring in the daimon of the proof π as a witness of an existential quantifer, this is managed through a substitution denoted $\leftarrow_\exists$, furthermore this steps add some dependencies to F_1.

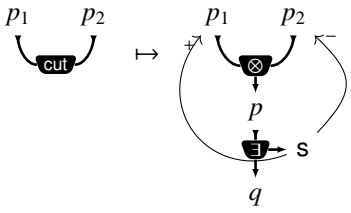


Fig. 21: Illustrating the substitution involved in Remark 41

Proof. Consider two links (e_1, e_2) which are subject to parsing denote $d(e_1)$ (resp. $d(e_2)$) the daimon(s) above the link e_1 (resp. e_2). Note that $d(e_1)$ and $d(e_2)$ contains either one daimon link or two, if $d(e_1) \cap d(e_2) = \emptyset$ (meaning the daimons above the links are distinct) confluence is straightforward: because the parsing of connectives replaces $d(e_1) + e_1$ with a new daimon link and edits the target of some pointers (the pointers whose target lies in the outputs of $d(e_1)$) as a consequence the two rewriting steps affect distinct part of the net and confluence is guaranteed.

We will therefore always assume that $d(e_1) \cap d(e_2)$ is not empty meaning the two links have at least one daimon link in common. We then treat the case one after the other (recall that the position in the sources of e_1 and e_2 can be target of $\forall$-ghosts or $\forall$-pointers and of $\exists$-ghost, however they cannot be the target of $\exists$-pointers);

- If e_1 is a parr link the confluence is obtained as in Figure 22
- If e_1 is an exists link the confluence is obtained as in Figure 23: in particular note that the existential pointer of the exists link must be a ghost for the rule to be applicable.
- If e_1 is a parr link the confluence is obtained as in Figure 24
- If e_1 is a tensor link while $d(e_1) \neq d(e_2)$ (this in particular that if both e_1 and e_2 are tensor links they cannot be placed under the same two daimons) the confluence is obtained as in Figure 25
- Finally assume that e_1 is a tensor link while $d(e_1) = d(e_2)$ in that case e_2 must also be a tensor, such a case is called the *critical pair*, we observe that confluence does not hold in this case, see Figure 26. However a net containing a critical pair is not a proof net because after one reduction of one of the two tensor (say e_1) the parsing reduction will never be able to reduce the other tensor link (in that case e_2) (Figure 26). Since S is supposed to be a proof net this case never happen.

□

Proposition 61. *The pointer rule defines a strongly confluent rewriting rule.*

Proof. This is illustrated in Figure 28 where the two existantial pointers are assumed to be in the same daimon link. □

Proposition 62. *Given a net S: $S \to^C_{\mathsf{P}} S_1$ and $S \to_{\mathtt{ptr}} S_2$ there exists S' such that $S_1 \to_{\mathtt{ptr}} S'$ while $S_2 \to^C_{\mathsf{P}} S'$*

Proof. The parsing $S \to^C_{\mathsf{P}} S_1$ of the connective link e implies that the sources of e cannot be the target of an active existantial link, hence in parrticular it is not the target of the existantial pointer subject to the rewriting $S \to_{\mathtt{ptr}} S_2$ (this is important to show confluence). The confluence diagrams are illustrated Figure 27. □

We illustrate the proof of confluence of the parsing reduction.

2) Existential Precedence and Kingdoms: We provide proofs of the remaining propositions of subsection VII-A.

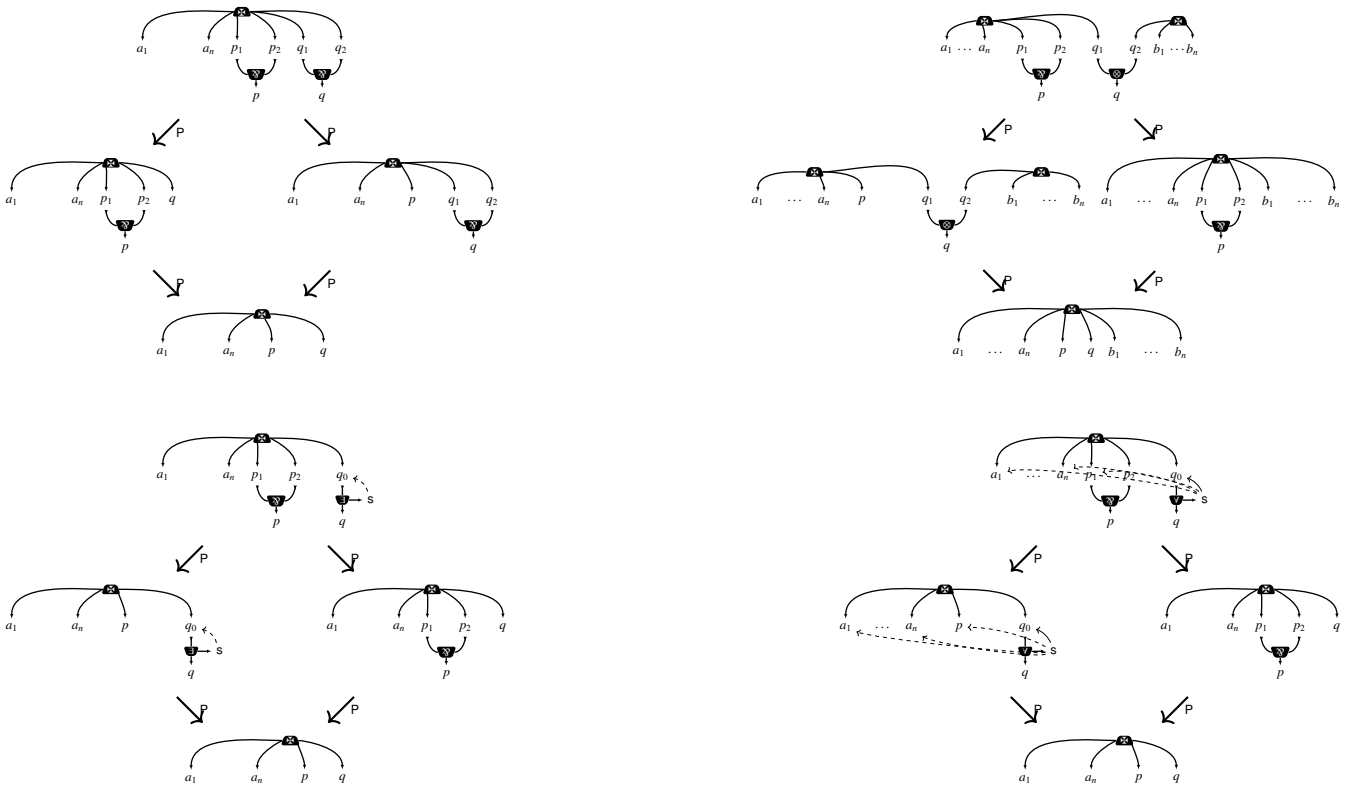


Fig. 22: Confluence cases of the form (⅋/?)

Lemma 63. *Given two nets $S \to_{\mathsf{P}}^{*} S'$; Each daimon link r in S' is associated with a sub-proof net P of S.*

Proof. We define inductively a sub proof structure P of S corresponding to r. We consider the reduction sequence from S to S' and its restriction s to the rules applied to elements of P (links or positions). Forgetting the pointers from $\exists$ and $\forall$ links in S outside of P, we have a proof structure P and a reduction s from P to a daimon. This means that P is correct, hence a proof net (by completeness). □

Remark 64. Given a net S and a reduction sequence $\vec{\alpha}$, let $\mathbf{p} = \langle \mathtt{s} \blacktriangleright_{\forall} t \rangle$ be some pointer in S, we can defined easily the pointer $\mathbf{p}[\vec{\alpha}]$ which is a pointer and the reduct of the pointer $\mathbf{p}$ after applying the reduction sequence $\vec{\alpha}$ to S. If $\mathbf{p}[\vec{\alpha}]$ targets a position p then for any sequence α_0 that is a prefix of α the pointer reduct $\mathbf{p}[\vec{\alpha}_0]$ targets a position p_0 that lies above p. In particular if $\mathbf{p}$ is a universal pointer while $\mathbf{p}[\vec{\alpha}]$ is a universal ghost it must be that the existential guard e of $\mathbf{p}$ has been reduced and that (the existential rule associated) with e is contained in the proof net associated with the daimon containing the target of $\mathbf{p}[\vec{\alpha}_0]$.

Proposition 46. *Given a net S and $\vec{\alpha}$ is a reduction sequence for S to a daimon, then for all $\forall$ link e in $\vec{\alpha}$, the daimon link produced by $\vec{\alpha}_{\leq e}$ is associated with a proof net containing a kingdom of the link e.*

Proof. After reducing S using the sequence $\vec{\alpha}_{\leq e}$ the rule for e creates a daimon link associated with a proof net P_e (Lemma 63). Furthermore, $\vec{\alpha}_{\leq e}$ contains the guards of e (Proposition 44).

Let $\mathbf{p} = \langle \mathtt{s} \dashrightarrow_{\forall} t \rangle$ be universal ghosts associated with e: since the rule on e can be applied, $\mathbf{p}$ must point at the daimon right above e, and the original universal pointer $\mathbf{p}'$ in S which as generated $\mathbf{p}$ must have its target above t (Remark 64). Since $\mathbf{p}'$ has become a ghost $\mathbf{p}$ it means that its guard has been contracted and since $\mathbf{p}$ points on the daimon above e it means that the guard of $\mathbf{p}'$ is contained in P_e the proof-net associated with the daimon e. By this argument all the guard of e are contained in the proof-net P_e.

Hence, P_e is a sub-proof net of S containing the guards of e: P_e therefore contains a kingdom for e. □

Remark 65. Not every weak reduction sequence which satisfied the existential precedence rule is a verification sequence, specifically because some reduction sequence might satisfy the existential precedence in S. but not apply to S; for instance this is the case of the net representing the second proof at the end of the section I. This is why the kingdom property was defined.

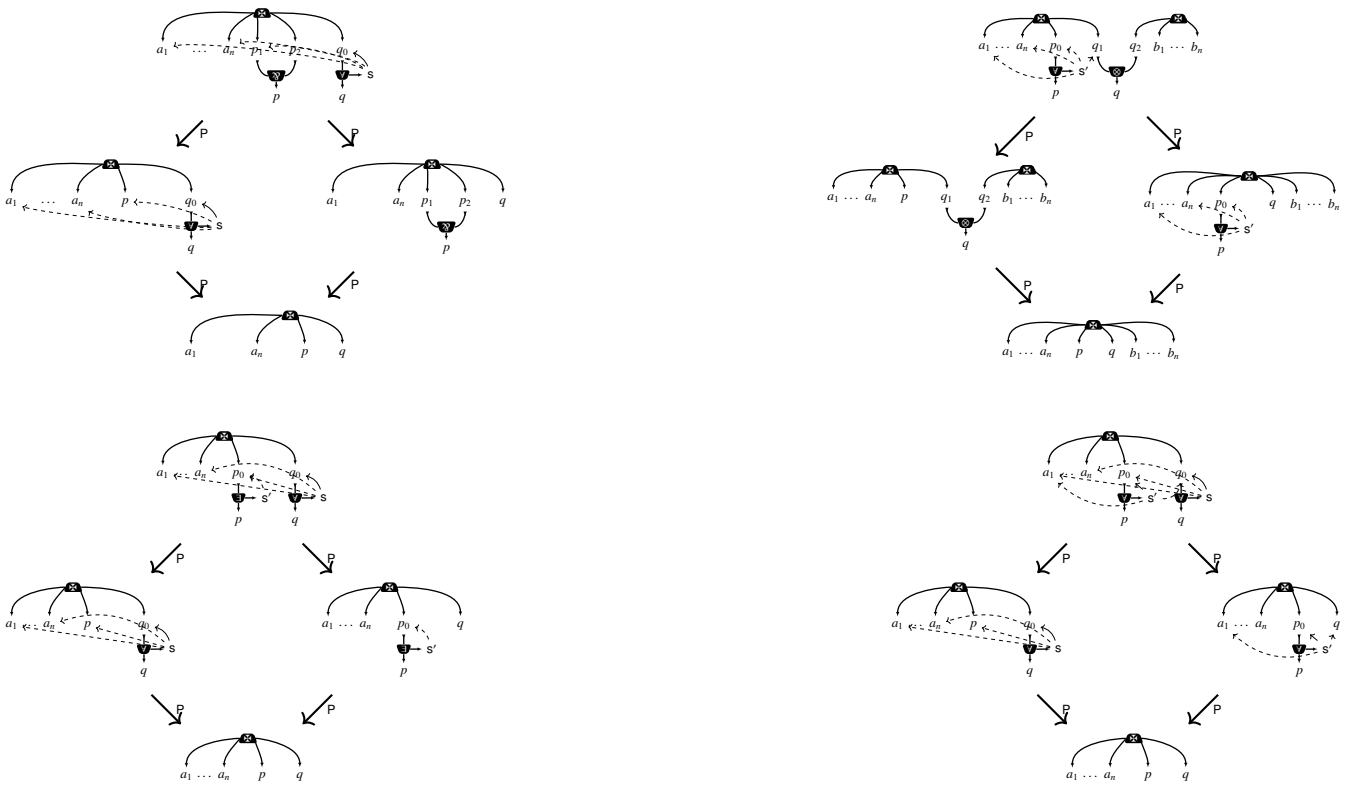


Fig. 23: Confluence cases of the form (∃/?)

Theorem 47. *Given a guarded net S (i.e. all its out of range universal pointers have a guard) and a full weak reduction sequence $\vec{\alpha}$ for S; if $\vec{\alpha}$ satisfies the kingdom property then $\vec{\alpha}$ is a verification sequence of S (hence S is a proof net).*

Proof. Since the weak reduction sequence $\vec{\alpha}$ is full for S if it is a reduction sequence of S then S is a proof net. The fact that the net is well scoped existentially ensures that the tensor, par, exists parsing rewriting can always be performed hence we must simply ensure that the forall rules can always be performed. Consider any for all link e in S: Indeed the condition ensure that when reducing $\vec{\alpha}_{\leq e}$ the daimon produced when reducing e contains one of its kingdom and thus all its guarding existsential link. This ensure that all the out of range pointer $\langle \mathsf{s} \blacktriangleright_\forall t \rangle$ have become ghosts $\langle \mathsf{s} \dashrightarrow_\forall t \rangle$ then the fact that the kingdom is located in a single daimon which lies above e is ensured by the property. Thus the universal ghosts of the forall link e points in the daimon just above e as a consequence the parsing rule can be applied. □

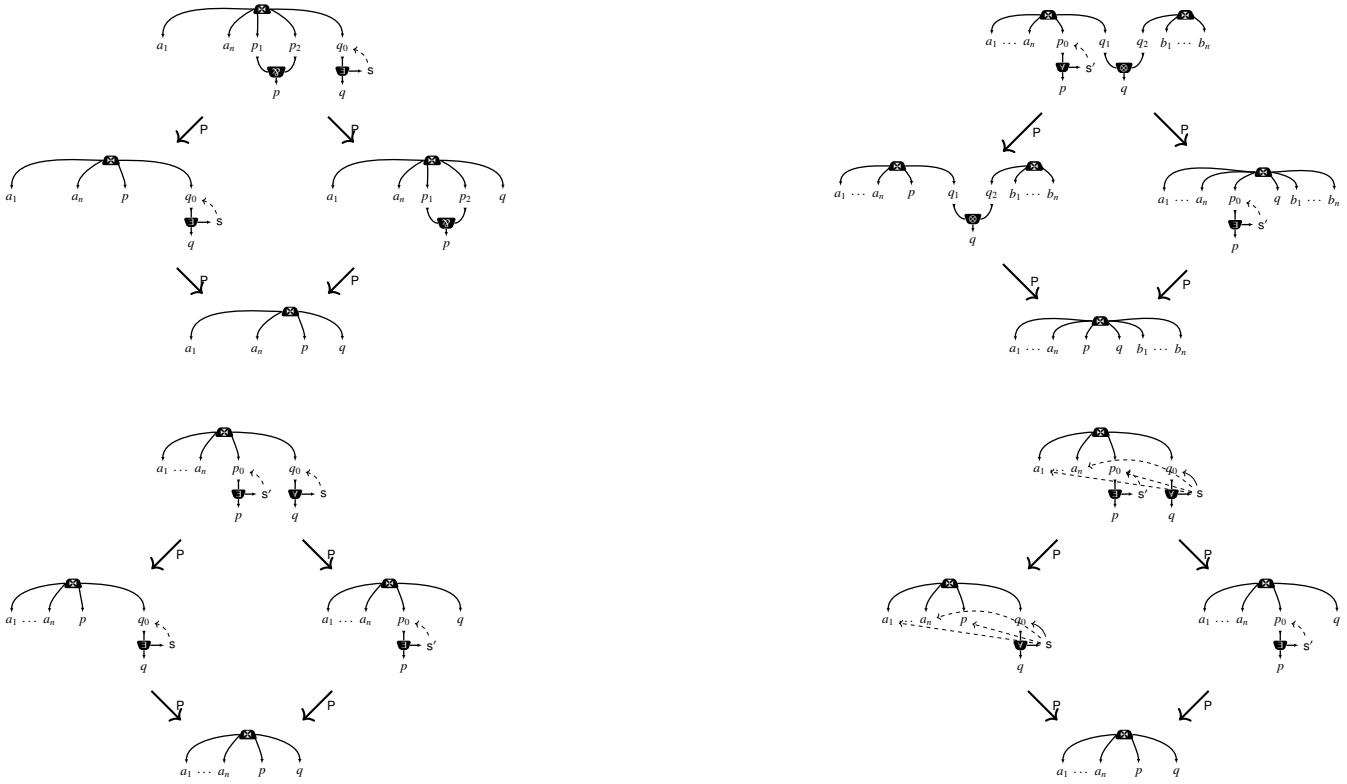


Fig. 24: Confluence cases of the form (∀/?)

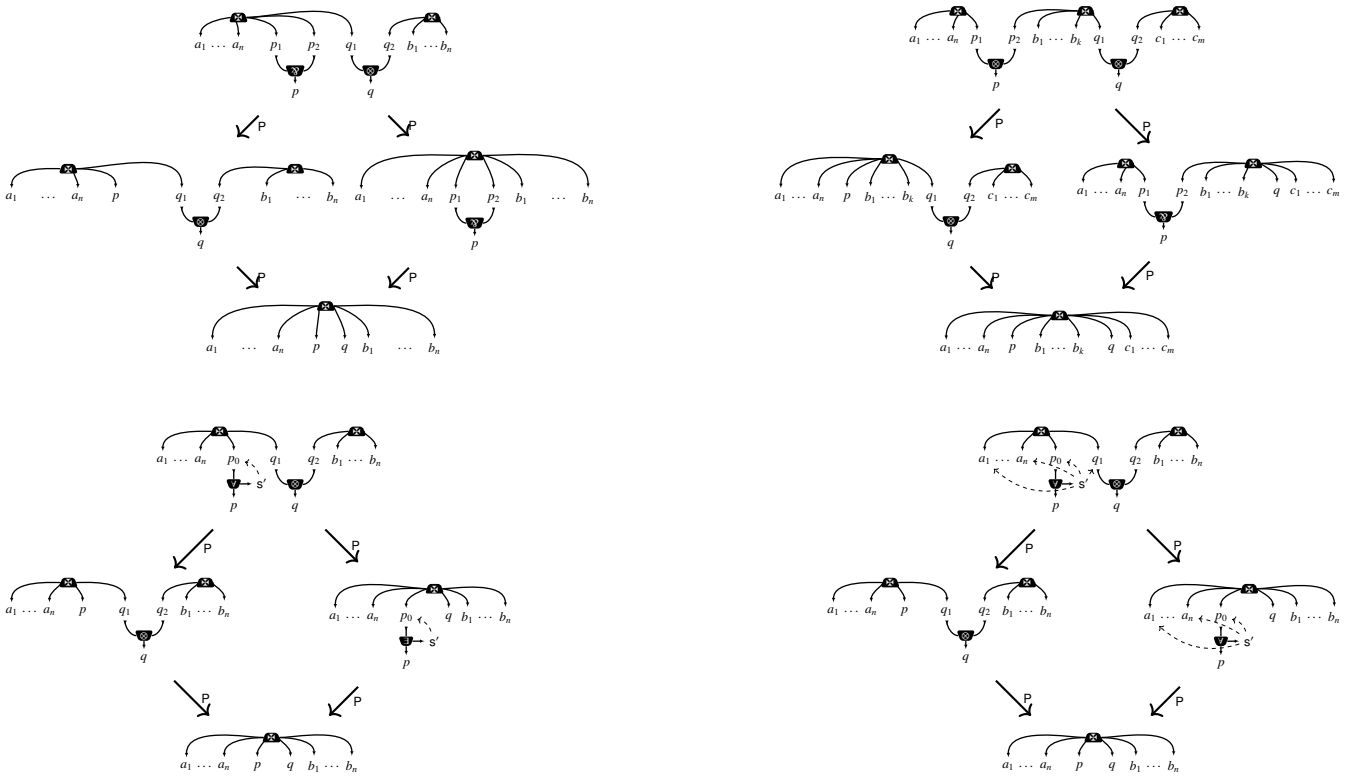


Fig. 25: Confluence cases of the form (⊗/?); the (⊗/⊗) case is not exhaustive and in particular does not illustrate the critical pair, the remaining (⊗/⊗) is treated in the next figure.

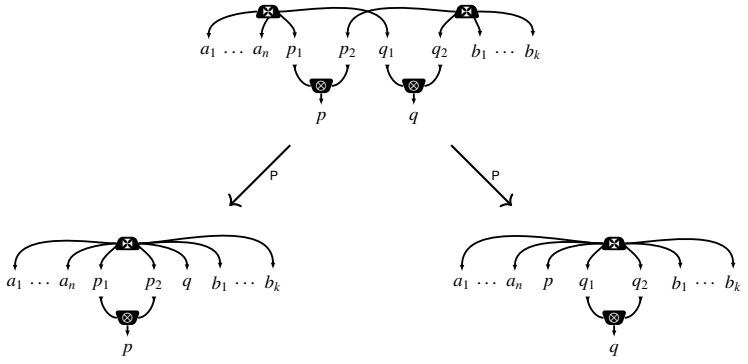


Fig. 26: Non confluence of the critical pair (⊗/⊗). In both reducts the remaining tensor is cyclic and thus cannot be contracted. Because we don't represent the context (other links occuring in the parsed net), the two redexes are isomorphic, however, if other links are involved the reducts can be distinct nets. The critical pair occurs only in incorrect nets, because (1) the two distinct reducts cannot contract the remaining tensor links; and (2) the parsing reduction may merge daimons but never split them thus an (incontractible) cyclic tensor below a daimon link can never become contractible after steps of parsing reduction.

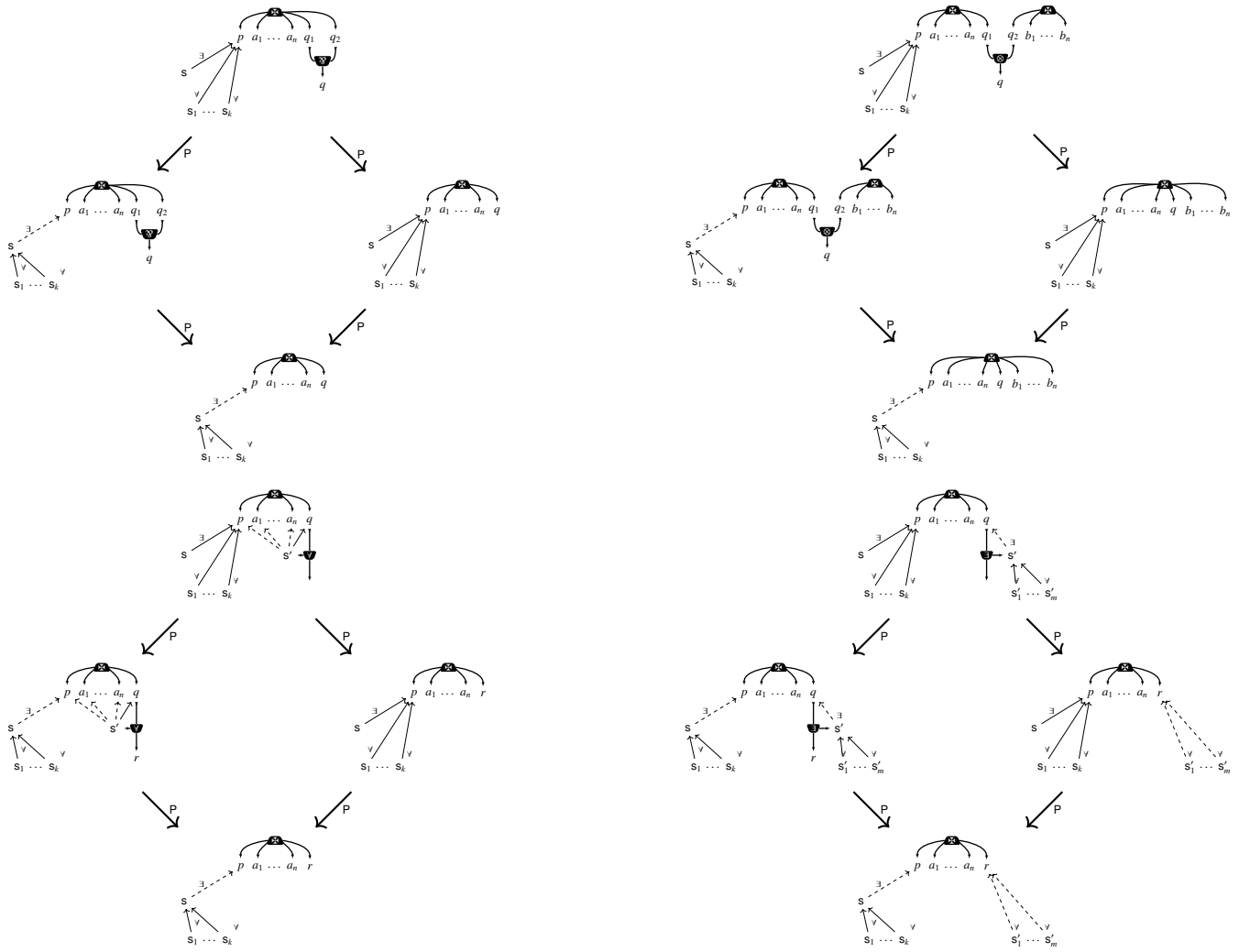


Fig. 27: Confluence cases of the form (reroute/?) where the other step is a parsing reduction for a connective; note that in the exists case the pointer port s and s′ are necessarily distinct because if the exists step can be performed the pointer of the exists link must be a ghost. Furthermore the position p and q must be distinct because q is the input of a contractible link, therefore it must be existential free.

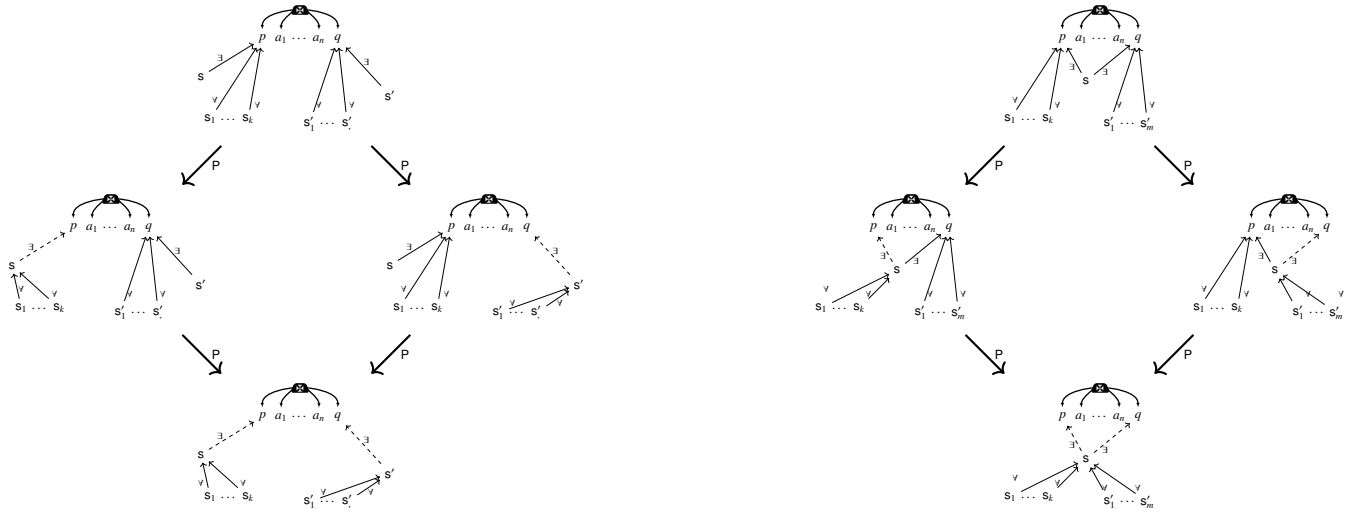


Fig. 28: Confluence cases of the form (reroute/reroute); we illustrate the two cases, when the two existential pointers have distinct source, and when both existential pointers have the same source.