Proofs as Execution Trees for the π-Calculus

Matteo Acclavio
University of Sussex

Giulia Manara
University of Southern Denmark, Universitè Paris Cité, Università Roma Tre

Abstract

In this paper, we establish the foundations of a novel logical framework for the π-calculus, based on the *deduction-as-computation* paradigm. Following the standard proof-theoretic interpretation of logic programming, we represent processes as formulas, and we interpret proofs as computations. To be precise, we interpret proofs as execution trees.

For this purpose, we define a cut-free sequent calculus for an extension of first-order multiplicative and additive linear logic. This extension includes a non-commutative and non-associative connective to faithfully model the prefix operator, and nominal quantifiers to represent name restriction. Finally, we design proof nets providing canonical representatives of derivations up to local rule permutations.

2012 ACM Subject Classification ; Theory of computation → Proof theory; Theory of computation → Linear logic; Theory of computation → Programming logic; Theory of computation → Process calculi

Keywords and phrases Non-commutative linear logic, Proof nets, Deduction-as-computation, Proof equivalence, π-calculus

Funding *Matteo Acclavio*: partially supported by the Villum Fonden grant no. 50079.
Giulia Manara: co-funded by the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 945332, and by the European Union (ERC, CHORDS, 101124225).

Acknowledgements Thanks to Giovanni Bernardi and Fabrizio Montesi for the insightful discussion about the process calculi (see [8])

1 Introduction

Formal reasoning about the properties of concurrent program executions is significantly more complex than analyzing sequential programs. The main challenge in the concurrent setting arises from the lack of formalisms for efficient representations of the set of traces of a program in the presence of *interleaving* concurrency, where the mutual order of certain tasks of a program is irrelevant. This is due to the inherent limitations of languages commonly used to represent trace reasoning, including natural language, where it can be impossible to describe a set of events arranged in complex patterns in a canonical way, other than by inefficiently listing all possible total orders. A language for optimizing the trace analysis of concurrent programs should:

1. ignore irrelevant differences, such as the mutual order of independent events;
2. group traces that differ only in branching caused by internal choices within the program;
3. distinguish sets of traces that differ due to factors beyond the control of the program, such as race conditions and side effects.

In this work, we develop a formalism satisfying these three desiderata to represent execution trees of processes of the π-calculus in a canonical way, based on the *deduction-as-computation* interpretation for proofs a logic extending first-order multiplicative additive linear logic.

An approach inspired by logic programming. In the *logic programming* paradigm, programs are interpreted as sets of formulas, and computation is performed by applying methods (or rules) to these formulas. In [65] Miller et al illustrate how a *deduction-as-computation* interpretation of proof search in the sequent calculus allows to account for program executions: sequents correspond to snapshots of the state of the system, and sequent rules can be interpreted as methods executing the instructions encoded by logical connectives.

In the setting of deduction-as-computation, two forms of non-determinism appear in program executions that are not as easily observable in other frameworks[1]: the *don't care* non-determinism, depending on the possibility of applying rules to independent sets of formulas, and the *don't know* non-determinism, that arises from the possibility of applying (potentially different) rules to overlapping subsets of formulas. In the proof-search interpretation of program execution, differences in derivations caused by don't care non-determinism are considered irrelevant, at the point that two proofs which can be transformed into one another through *rule permutation* (i.e., by exchanging the order of rules operating on disjoint sub-sequents) are usually identified.

This work aims to apply results in the study of proof equivalence [50, 49, 52, 82] in the framework of deduction-as-computation to provide canonical representations of sets of traces. In particular, we develop a syntax based on results about canonical representation of proofs to uniquely model a set of traces differing in the order of independent events, in compliance with desiderata 1. We focus on a deduction-as-computation interpretation of *proof nets* rather than sequent calculus derivation.

Why proof nets? Various works [14, 13, 7] have already highlighted the benefits of this approach where the syntax captures interleaving concurrency by default. Proof nets were introduced as a graphical formalism for *linear logic* proofs [33]. They abstract away irrelevant information contained in sequent calculus derivations, such as mutual order of independent inference rules. This syntax allows for an optimal level of abstraction for the multiplicative fragment of linear logic (MLL), providing canonical representatives for proofs with respect to independent rule permutations, a polynomial proof translation, and a geometrical correctness criterion allowing to check in polynomial time if a graph is the encoding of a proof (making proof nets for MLL a proof system in the sense of [24]). However, the definition of proof nets for extensions of MLL requires trade-offs between canonicity, the efficiency of correctness criterion and the efficiency of normalization procedure (see [41] for MLL with units, [55, 56, 53, 52] for multiplicative-additive linear logic (MALL) and [2] for multiplicative-exponential linear logic).

To provide an intuition of our approach, we show for the process P in Equation (1) how we can annotate information about communications (and selections) performed during all the possible executions of P while ignoring inessential details such as the specific order of independent transitions.

$$P = (\nu x)(\nu y)\Big(y!\langle a\rangle \mid y?(a) \mid x \triangleright \big\{ \ell_1 : x?(b) , \ell_2 : x!\langle c\rangle \big\} \mid x \triangleleft \big\{ \ell_1 : x!\langle b\rangle , \ell_2 : x?(c) \big\}\Big) \quad (1)$$

Note that the only (don't know) non-determinism during executions of P is caused by an internal choice: the branching due to the choice of a label in $L = \{\ell_1, \ell_2\}$. Thus, we have a unique proof net according to the desiderata 1 and 2. In this representation the set of links $\{\ell_1, b\}$ and $\{\ell_2, c\}$ are mutually exclusive – in the terminology of event structures [86], we would say they are in conflict relation.

[1] In particular, within the proofs-as-processes setting that arises from the Curry-Howard isomorphism, the non-determinism resulting from an internal choice is not observable in the computations of typed processes, as the type of a process predetermines the choice.

At the same time, our syntax still allows us to distinguish the two distinct executions of the process in Equation (2), as specified by desideratum 3, which are determined by a race condition on x enforcing a (don't know) non-deterministic choice during the execution.

$$(\nu x)\left(x!\langle a\rangle.\ x!\langle b\rangle \mid x?(y) \mid x?(z)\right) \quad \text{and} \quad (\nu x)\left(x!\langle a\rangle.\ x!\langle b\rangle \mid x?(y) \mid x?(z)\right) \qquad (2)$$

Which proof nets? There are different syntaxes for proof nets for MALL, each providing a different level of abstraction and capturing different subsets of the rule permutations in the sequent calculus – rules permutations are reported in Figure 16. The first syntax of proof nets for MALL was introduced by Girard in [33]. These are referred to as *box nets* because of the way they encode the rules for the additive conjunction &. These proof nets provide polynomial time proof translation and correctness criterion. However, they are not canonical with respect to rule permutations involving the rule for &. In [36], Girard also introduced *monomial nets* for MALL, lacking of a polynomial time correctness criterion (but a polynomial-time proof translation), but unable to not improve the level of abstraction in the general setting. For this reason, Hughes and Van Glabbeek introduced in [55, 56] a new syntax of proof nets for MALL called *slice nets*. These proof nets capture all rule permutations in Figure 16 while keeping a polynomial correctness criterion, but they lack of a polynomial proof translation. The absence of a polynomial proof translation is unsurprising because each slice net can be conceived as a canonical representative of a class of derivations, which may include derivations whose sizes differ by an exponential factor. This is due to the fact that the rule permutation between the two-premises multiplicative rule $\otimes$ and the two-premises additive rule &-rules requires to duplicate an entire subtree of the derivation (see Figure 16). In order to recover a polynomial proof translation, Hughes and Heijltjes introduced *conflict nets* in [53, 52], which can capture only *local* rule permutations – that is, all rule permutations except the one between the & and the $\otimes$. This is obtained by having trees of axiom links (instead of a set of sets of axiom links as in slice nets) where axiom links are in a "multiplicative" concordance relation, or in a "additive" conflict relation.

At the same time, there are two main approaches for proof nets for first-order logic. One is to consider the choice of witnesses for the existential quantifier (and, in our case, for the nominal quantifiers) as part of the information of a proof. This leads to the notion of what we refer to as *witness nets* (as in [42]) developed in [33, 34, 35], which in [37, Chapter 11] Girard claims to be "the only really satisfactory extension of proof-nets". However, how explained by Hughes in [51], having the witnesses being part of the proof leads to undesirable consequences such as the lack of canonical proofs due to the existence of infinitely many possible witnesses (possibly of exponentially larger size); for an example, consider the formula $\exists P.(x) \multimap \exists P.(x)$ which has infinitely many proofs, one for each possible witness for x. Moreover, the complexity of cut-eliminination becomes exponential, and requiring non-local rewriting rules even in first-order MLL. For this reason, in [51] Hughes introduced another approach for the design of proof nets for first-order logic, which abstracts away the choice of witnesses for the quantifiers, thus satisfying the principle of *generality* (in the sense of Lambek [58]) by identifying those proofs differing in the witness assignment. These proof nets, called *unification nets*, were initially developed for first-order MLL, and have been recently extended for the purely additive fragment of linear logic in by Hughes, Heijltjes and Straßburger in [42]).

In this paper we develop both conflict and slice (unification) nets for PiL to provide canonical representatives of execution trees for the π-calculus modulo different notions of interleaving:

- Conflict nets (Section 5.1) abstract away the order of independent transitions, as soon as this order does not interact with branching of the execution tree.

- Slice nets (Section 5.2) abstract away the order of any independent transitions in a process, even when the execution tree branches.

In choreographic programming [70], the distinction between these two flavors of interleaving can be easily understood in terms of restriction on the possibility of performing out-of-order instruction in the operational semantics: while slice nets identify execution trees of the standard operational semantics, where both communications and choices can be delayed, conflict nets allow to only delay communications, and not choices.

Contributions of the Paper. We develop a *new* logical framework (PiL) based on an extension of first-order multiplicative and additive linear logic (MALL^1) with a *non-commutative non-associative connective* and *nominal quantifiers*, to provide logical operators that faithfully model the high-level search instruction corresponding to the prefix composition and restriction in the π-calculus. We define a cut-free sequent calculus in which execution trees of a process can be interpreted as derivations of the corresponding formula.

We also define two syntaxes of proof nets for PiL by combining the techniques used in *unification nets* for first-order multiplicative linear logic [54] and first-order additive linear logic [42], with the techniques used in *conflict nets* [53, 41] and in *slice nets* [56] for MALL. As a restriction of our syntax, we then define unification conflict nets and unification slice nets for first-order MALL.

Finally, by combining the correspondence between execution trees and derivations, and between derivations modulo local rule permutations and proof nets, we provide a syntax in which we have canonical representatives of execution trees modulo interleaving.

Related Works on Processes as Formulas. Following the ideas in [65], Miller proposed in [64] a theory within linear logic allowing to interpret the reduction semantics of the π-calculus as implication in the theory, where parallel is internalized by the $\parr$ and the choice operator $+$ in the original formulation of the π-calculus [69] by the $\oplus$. Guglielmi developed an extension of multiplicative linear logic with a non-commutative connective aiming at internalizing sequentiality in [38, 39], lately leading to the design *deep inference* and the formalism of the *calculus of structures* [40] to obtain a satisfactory proof system for the logic BV. In [21] Bruscoli established a *processes-as-formulas* and *computation-as-deduction* correspondence for a simple fragment of CCS (without recursion, choice, and restriction) where each successful terminating execution of a process corresponds to specific derivations in BV. This correspondence has been extended to the π-calculus by Horne, Tiu et al. [45, 46, 48], including the choice operator ($+$), modeled via the additive connective $\oplus$, and name restriction, modeled using *nominal quantifiers* in the spirit of [73, 30]. We highlight here the main differences of our approach with respect to the aforementioned works:

- we use a non-associative non-commutative self-dual connective $\blacktriangleleft$ (instead of the non-commutative but associative $\triangleleft$ in BV). This choice allows for a cut-free sequent calculus for PiL, while no sequent calculus for BV or any of its extension can exist [83];
- in [47, 48] Horne et al. use the original version of the π-calculus [69] which feature a choice operator $+$ with an undesirable "non-local" behavior, which requires to forwardly check that it will entail a communication rule. Its rule is written as follows

  $$+ : A + B \quad \rightarrow \quad A' \qquad \text{only if } A \rightarrow A'$$

  That is, the choice operator $+$ is not complitely free to choose between A and B, but it is constrained by the possibility of performing an action after such a choice. That is, if A cannot perform any action, then the choice $A + B$ cannot reduce to A.

  A logical operator modeling such a choice operator should have a rule capable of spotting (within a given context) the sub-formulas on which some rules can be applied. Such a

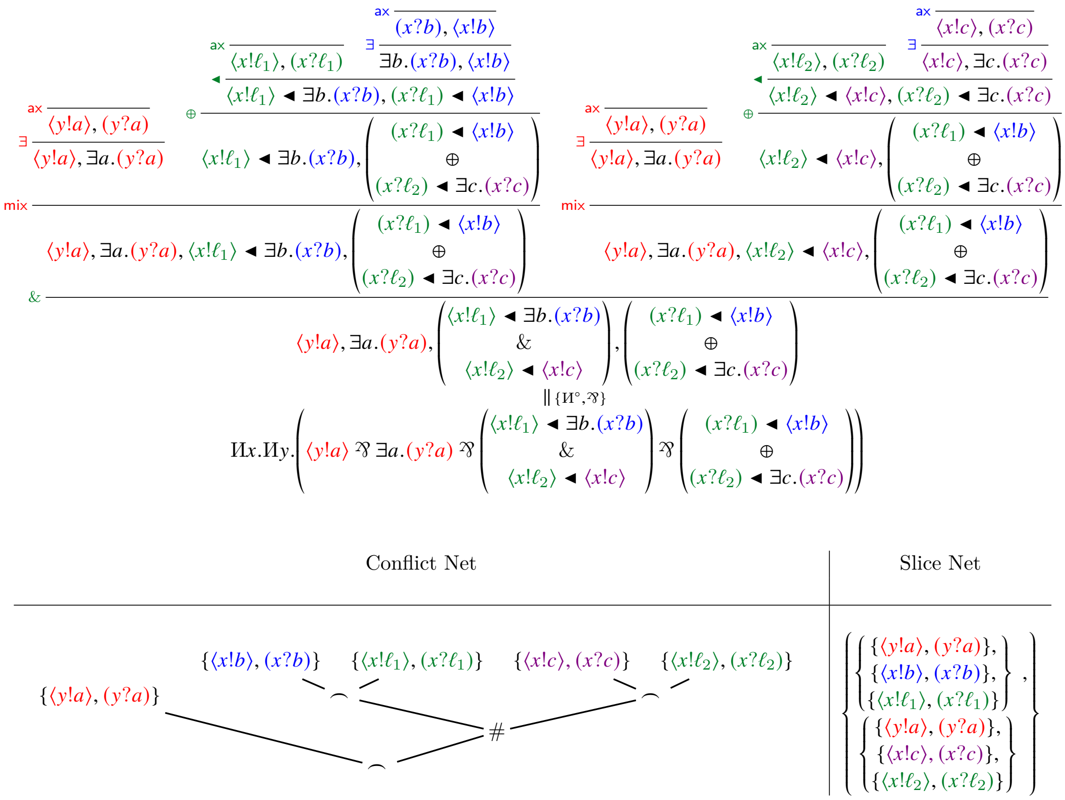


Figure 1 A derivation of the formula $[\![P]\!]$ from Equation (1) corresponding to execution tree in the right of Figure 4, and its corresponding conflict net.

behavior, to the best of our knowledge, has never been studied in the literature of proof theory. For this reason, we consider the version of the π-calculus from [85, 32] in which the two choice operators play different roles: the *label-send* $x \triangleleft \{\ell : P_\ell\}_{\ell \in L}$ allows a process to choose its continuation independently of the environment (which we model with the additive conjunction &, whose rule branches a derivation duplicating the context), while the *label-receive* $x \triangleright \{\ell : P_\ell\}_{\ell \in L}$ allows a process to choose according to the environment (which we model with the additive disjunction $\oplus$, whose rule is applied according to the context's need). This latter version of the π-calculus is the one used in the literature of session types [85, 32, 44] and choreographic programming [70].[2]

- in the work of Bruscoli [21], and in the works of Horne and Tiu [48, 46] derivations correspond to executions, while in our work derivations represent execution trees. In the latter works, the original Milner's choice operator $(+)$ is modelled in the system MAV by using the additive connective $\oplus$ from additive linear logic, since the additional information about the environment seems to be guaranteed (a-posteriori) by the fact that the correspondence is only established between successful executions and derivations. However, undesired behaviors may still occur in establishing the correspondence executions-as-derivations in MAV. For an example, consider the process $A + \mathsf{Nil} \mid x!\langle a\rangle \mid x?(a).$, which

[2] Applications of the logical framework we develop, as well as more precise connections with session types and choreographic programming are presented in the companion paper [8].

is stuck, but derivable in MAV.
Note that if we restrict the label-send constructor $x \triangleleft \{\ell : P_\ell\}_{\ell \in L}$ on singleton sets of labels, or, equivalently, if we redesign the rule for the additive connective & in such a way it only has a single premise (that is, we prune the other premise), then we can recover a correspondence between derivations and executions also in our setting.

- as Horne and Tiu in [45, 46, 48], we use a pair of dual nominal quantifiers (instead of a self-dual quantifier as in [63, 67, 78]) to model restriction.[3] However, as explained in detail in Remark 8 and in Appendix A, our pair of dual quantifiers satisfies different proof theoretical properties.

Structure of the paper. In Section 2 we recall standard definitions for sequent systems and syntax and semantics of the π-calculus. In Section 3 we present formulas and sequent systems, explaining the design choices we made in the opeartors of the logic PiL. In Section 4 we study their proof theoretical properties of our system, including relevant formula equivalences and cut-elimination. In Section 5 we present the syntax of proof nets for PiL, providing translations from derivations to proof nets, and from proof nets to derivations (sequentialization). In Section 6 we prove canonicity for our proof nets with respect to local rule permutations. In Section 7 we show how formulas in PiL can be used to encode processes of the π-calculus, and how execution trees of a process P can be represented by derivations of the corresponding formula. Thereby, we show that equivalent execution trees (modulo interleaving) can be represented by the same proof net. We conclude in Section 8 by discussing extensions of this framework and its possible applications.

2 Preliminary Notions

We assume the reader to be familiar with the notion of trees and of formula tree, as well as with the syntax of sequent calculus (see, e.g., [84]), but we recall here the main definitions. We may identify formulas with their formula-trees and we consider ***sequents*** as forests made of formula-trees.[4] of formulas in a given grammar.

A ***sequent rule*** r is an expression of the form $\mathsf{r}\frac{}{\mathcal{S} \vdash \Gamma}$, $\mathsf{r}\frac{\mathcal{S} \vdash \Gamma_1}{\mathcal{S} \vdash \Gamma}$, or $\mathsf{r}\frac{\mathcal{S} \vdash \Gamma_1 \quad \mathcal{S} \vdash \Gamma_2}{\mathcal{S} \vdash \Gamma}$. The sequent Γ is called ***conclusion*** and the sequents above the line ***premises***. An occurrence of formula in the conclusion (resp. in a premise) of a rule but in none of its premises (resp. not in its conclusion) is said ***principal*** (resp. ***active***). A ***sequent system*** X is a set of sequent rules.

A ***derivation*** in X is a non-empty tree $\mathcal{D}$ of sequents, whose root is called ***conclusion***, such that each sequent in $\mathcal{D}$ is conclusion of a rule in X, whose children are (all and only) the premises of the rule. An ***open derivation*** is a derivation whose leaves may be the conclusion of no rules, in which case are called ***open premises***. We may denote a derivation (resp. an open derivation with an open premise Δ) $\mathcal{D}$ with conclusion Γ by $\begin{array}{c}\mathcal{D}\Vert\\ \mathcal{S} \vdash \Gamma\end{array}$ $\left(\text{resp. } \begin{array}{c}\mathcal{S} \vdash \Delta\\ \mathcal{D}\Vert\\ \mathcal{S} \vdash \Gamma\end{array}\right)$.

[3] In [45] the authors report the use of a non-self-dual quantifier to model restriction was suggested them by Alessio Guglielmi in a private communication.
[4] Said differently, a sequent is a set of occurrences of formulas. Note that defining a sequent as a multiset of formulas would require the introduction of additional structure to pinpoint on which occurrences of formulas rules are applied, making way more cumbersome the definition of proof nets (Section 5) and preventing the confluence of cut-elimination due to the impossibility of distinguishing which occurrence of formula is active for a cut.

Processes		Structural Equivalence
$P, Q, R \coloneqq \mathsf{Nil}$	nil	$P \equiv P^{\alpha}$
$\mid x!\langle y\rangle.P$	send (y on x)	$P \mid Q \equiv Q \mid P$
$\mid x?(y).P$	receive (y on x)	$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
$\mid P \mid Q$	parallel	$(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$
$\mid (\nu x)P$	restriction (or nu)	$P \mid \mathsf{Nil} \equiv P$
$\mid x \triangleleft \{\ell : P_\ell\}_{\ell \in L}$	label-send (on x)	$(\nu x)S \equiv S$
$\mid x \triangleright \{\ell : P_\ell\}_{\ell \in L}$	label-receive (on x)	$(\nu x)P \mid S \equiv (\nu x)(P \mid S)$
with $x, y \in \mathcal{N}$ and $L \subset \mathcal{L}$. The constructors binding variables are $(\nu x)P$ binding x in P, and $x?(y).P$ binding y in P only		P^{α} α-equivalent to P; x is not a name occurring free in S

Reduction Semantics

Com:	$x!\langle a\rangle.P \mid x?(b).Q \to P \mid Q[a/b]$		Res :	$(\nu x)P \to (\nu x)P'$	if $P \to P'$
Bra:	$x \triangleleft \{\ell : P_\ell\}_{\ell \in L} \to x \triangleleft \{\ell_k : P_{\ell_k}\}$	if $\ell_k \in L$	Par :	$P \mid Q \to P' \mid Q$	if $P \to P'$
Sel:	$x \triangleleft \{\ell : P_{\ell_k}\} \mid x \triangleright \{\ell : Q_\ell\}_{\ell \in L} \to P_{\ell_k} \mid Q_{\ell_k}$	if $\ell_k \in L$	Struc :	$P \to Q$	if $P \equiv P' \to Q' \equiv Q$

Figure 2 Syntax for processes, the relations generating the structural equivalence ($\equiv$), and the reduction semantics of the π-calculus. The α-equivalence is defined in the usual way (see Appendix).

2.1 π-Calculus

We consider the version of π-calculus presented in [85, 32], whose processes are generated from a countable set of ***(channel) names*** $\mathcal{N} = \{x, y, \ldots\}$ and (disjoint) finite set of ***labels*** $\mathcal{L}$ grammar in Figure 2. [5] In the same figure, we recall the definition of the ***structural equivalence*** ($\equiv$), as well as the ***reduction semantics***. We write $P \not\equiv Q$ if $P \equiv Q$ does not hold. We may denote by $\mathcal{N}[P]$ a process of the form $(\nu x_1)\cdots(\nu x_n)(P \mid Q)$ for some names $x_1, \ldots x_n$ and a process Q, and write a instead of $a.\mathsf{Nil}$ if $a \in \{x!\langle y\rangle, x?(y)\}$. We denote by $\twoheadrightarrow$ the transitive closure of $\to$.

A process P is ***stuck*** if $P \not\equiv \mathsf{Nil}$ and there is no P' such that $P \to P'$. A process P is called ***deadlock-free*** if P is not stuck and there is no stuck process P' such that $P \twoheadrightarrow P'$. A process P is ***race-free*** if there is no P' such that $P \twoheadrightarrow P'$ for a P' structurally equivalent to one of the following processes:

$$\begin{array}{ll} \mathcal{N}[(x!\langle y\rangle.R \mid x!\langle z\rangle.Q \mid S)] & \mathcal{N}[(x \triangleleft \{P_\ell\}_{\ell \in L} \mid x \triangleleft \{P_\ell\}_{\ell \in L'} \mid S)] \\ \mathcal{N}[(x?(y).R \mid x?(z).Q \mid S)] & \mathcal{N}[(x \triangleright \{P_\ell\}_{\ell \in L} \mid x \triangleright \{P_\ell\}_{\ell \in L'} \mid S)] \end{array} \quad (3)$$

A ***execution trees*** of a process P is a trees of processes with root P, where a process Q' is a child of Q if $Q \to Q'$, and such that branching is determined by the intrinsic non-determinism of the reduction rule Bra, that is, if two processes Q_1 and Q_2 are children of a same process Q, then $Q \to Q_1$ and $Q \to Q_2$ via Bra applied to the same minimal (w.r.t. term inclusion) sub-process of Q. We may label the edges of a execution tree with the unique reduction rule in $\{\mathsf{Com}, \mathsf{Bra}, \mathsf{Sel}\}$ required to reduce the term P to Q.[6] The ***interleaving*** equivalence relation ($\sim$) on execution trees is defined by the relations in Figure 3. See Figure 4 for an example of two execution trees equivalent modulo interleaving.

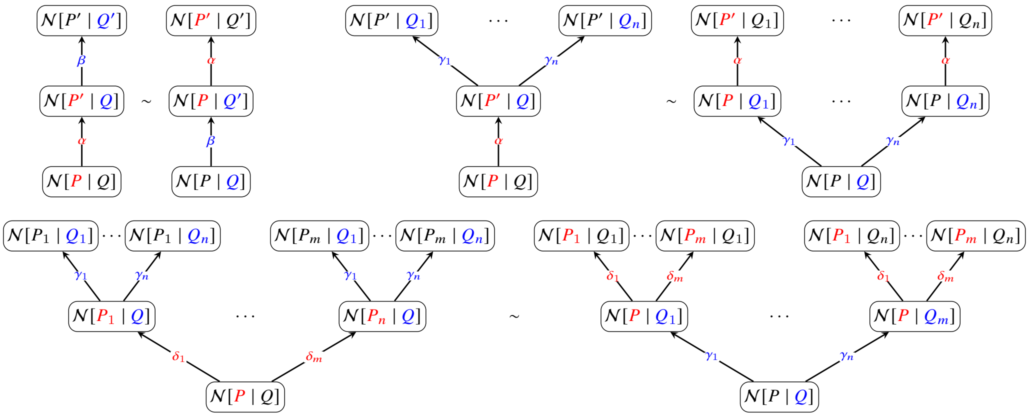
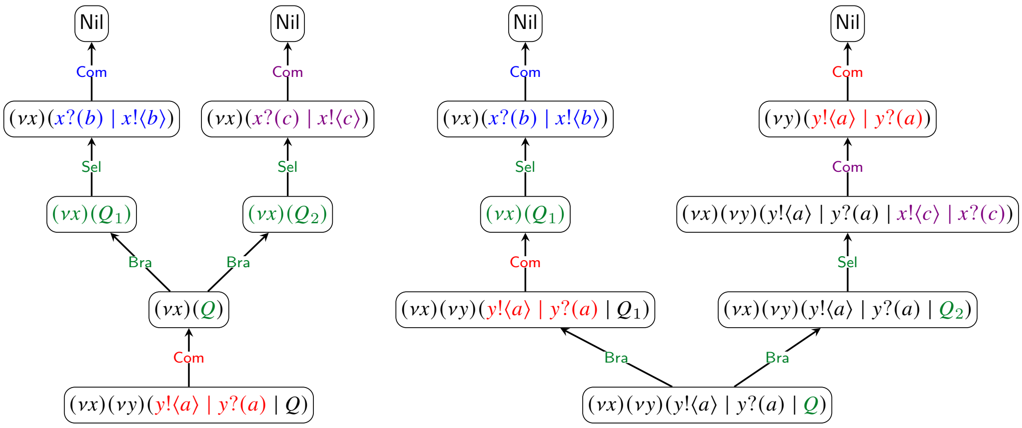


Figure 3 Generators of the execution tree equivalence with $\alpha, \beta \in \{\mathsf{Com}, \mathsf{Sel}\}$ and $\gamma_i, \delta_j \in \{\mathsf{Bra}\}$, where $\{P_1, \ldots, P_m\}$ (resp. $\{Q_1, \ldots, Q_n\}$) is the set of all processes such that $P \to P_i$ (resp. $Q \to Q_j$) via Bra.



where $Q_1 = x \triangleleft \{\ell_1 : x?(b)\} \mid x \triangleright \{\ell_1 : x!\langle b\rangle, \ell_2 : x?(c)\}$ and $Q_2 = x \triangleleft \{\ell_2 : x!\langle c\rangle\} \mid x \triangleright \{\ell_1 : x!\langle b\rangle, \ell_2 : x?(c)\}$

Figure 4 Two equivalent execution trees of the process P from Equation (1).

3 A New Logical Framework for the π-calculus

In this section we construct proof systems extending *first-order multiplicative additive linear logic* (or MALL[1]) with new operators allowing us to fitfully capture the behavior of term constructors for processes of the π-calculus w.r.t. the reduction semantics.

For this purpose, we enrich the language of MALL[1] with a non-commutative connective $\blacktriangleleft$ designed to capture the logical properties of the (non-commutative) prefix operator used in the π-calculus [69] (but also in CCS [68]). Even if it would be desirable to require $\blacktriangleleft$ to be associative, to capture the associativity of sequential composition of processes, we instead let

[5] As standard, may write $x!\langle y\rangle$ (resp. $x?(y)$) instead of $x!\langle y\rangle.\mathsf{Nil}$ (resp. $x?(y).\mathsf{Nil}$).

[6] The reduction rules Res, Par and Struc are not “meaningful” with respect to the computation, and even if a transition step may require multiple instances of these rules to deal with the bureaucracy of the syntax and the structural congruence, only a single instance of a rule in $\{\mathsf{Com}, \mathsf{Bra}, \mathsf{Sel}\}$ is required to perform a reduction step. For a formal definition of the labelling of the execution tree, see the definition of *core-reduction* in [8].

Formulas		
$A, B :=$	$\circ$	unit (atom)
$\mid$	$\langle x!y \rangle$	atom
$\mid$	$(x?y)$	atom
$\mid$	$A \parr B$	par
$\mid$	$A \otimes B$	tensor
$\mid$	$A \blacktriangleleft B$	prec
$\mid$	$A \oplus B$	oplus
$\mid$	$A \,\&\, B$	with
$\mid$	$\forall x.A$	for all
$\mid$	$\exists x.A$	exists
$\mid$	Иx.A	new
$\mid$	Яx.A	ya

De Morgan Laws		
$\circ^\perp$	$=$	$\circ$
$(A^\perp)^\perp$	$=$	A
$\langle x!y \rangle^\perp$	$=$	$(x?y)$
$(A \parr B)^\perp$	$=$	$A^\perp \otimes B^\perp$
$(A \blacktriangleleft B)^\perp$	$=$	$A^\perp \blacktriangleleft B^\perp$
$(A \oplus B)^\perp$	$=$	$A^\perp \,\&\, B^\perp$
$(\forall x.A)^\perp$	$=$	$\exists x.A^\perp$
(ИX.A)$^\perp$	$=$	ЯX.A$^\perp$

α-equivalence
$a = a$ if $a \in \{\circ, \langle x!y \rangle, (x?y)\}$
$A_1 \odot A_2 = B_1 \odot B_2$ if $A_i = B_i$ and $\odot \in \{\parr, \blacktriangleleft, \otimes, \oplus, \&\}$
Ꝺx.A = Ꝺy.A[y/x] y fresh for $A$and Ꝺ ∈ {И, Я, ∀, ∃}

Figure 5 Formulas (with $x, y \in \mathcal{V}$), and their syntactic equivalences.

$\blacktriangleleft$ being non-associative to reflect the fact that the prefix operator only allows prefixing a single atomic action at a time, and thus it does not model sequential composition because unable to compose sequentially non-atomic processes.

To capture restriction, following the spirit of the nominal quantifiers as introduced in [30], we use the nominal quantifier И allowing variable binding. As already explained in [67], the universal quantifier cannot not be used to satisfactorily model restriction. For an example, consider the processes $Q = (\nu x)(\nu y)(\mathsf{Nil}!\langle x \rangle.a \mid \mathsf{Nil}?(y).a)$ and $R = (\nu z)(\mathsf{Nil}!\langle z \rangle.a \mid \mathsf{Nil}?(z).a)$: if we encode restriction by universal quantification, then any property for Q should also be valid for R, because $\forall x.\forall y.P(x, y)$ entails $\forall z.P(z, z)$. The use of the existential quantifiers to model the variable binding of the input action, and nominal quantifiers for restriction allow us to avoid an unsound semantical overlap due to the duality of existential and universal quantification. For example, if restriction were modelled by the universal quantifier, then the deadlocked process $(\nu a)x!\langle a \rangle.\mathsf{Nil} \mid x?(a).\mathsf{Nil}$ would be encoded by the tautology $(\forall a.(\langle x!a \rangle \blacktriangleleft \circ)) \parr (\exists a.((x?a) \blacktriangleleft \circ)) = (\exists a.((x?a) \blacktriangleleft \circ)) \multimap (\exists a.((x?a) \blacktriangleleft \circ))$, while Theorem 47 shows that the formula encoding the process should not be provable. This mismatch is due to the translation, which allows unsound interactions between the binding of input actions with binding of restriction via a duality which is not valid in the semantics.

▶ Remark 1. In our work, we do not consider a self-dual nominal quantifier as the one studied in [67, 23, 78], but we rather introduce a dual quantifier similarly to what is done in [45, 46, 48], where such a design choice is justified in view of the semantics of the π-calculus.

In [48] the authors list the three following logical properties a nominal quantifier Ꝺ modeling the binder should satisfy:

1. *equivariance*, Ꝺx.Ꝺy.A and Ꝺy.Ꝺx.A should be logically equivalent;
2. *non-diagonality*: the formula Ꝺx.Ꝺy.A(x, y) should not imply Ꝺz.A(z, z) or vice versa;
3. *scope extrusion*: if $\ominus$ is a connective modeling parallelism, then (Ꝺx.A) $\ominus B$ implies Ꝺx.$(A \ominus B)$ whenever x does not occur in B.

In our work, all these requirements are met (see Proposition 11), but we also claim that the following additional condition should be added in this list, in view of how restriction and choice operators interact.

4 *name-choice*: if $\odot$ is a connective modeling choice, then Ꝺx.A $\odot$ Ꝺx.B should imply Ꝺx.$(A \odot B)$.

Intuitively, this latter requirement is dictated by the observational indistinguishably of a process spawning a fresh name before making a global choice, and a process spawning a fresh name after such a choice is made.

Note that in [73] this latter condition is required to hold for all connectives, and in [48] such condition holds for the sequential operator. However, such a behavior may not be desirable in certain contexts where two processes executed in sequence may behave differently if they share a communication channel or not. For an example, consider the case in which a communication channel become loose or vulnerable to attacks after events depending on the use of the channel.

► **Definition 2.** ***Formulas*** *are generated by a countable set of* ***variables*** *($\mathcal{V}$) by the grammar in Figure 5 modulo the standard* ***De Morgan Laws*** *and* ***α-equivalence*** *from the same figure. A* ***context*** *is a formula containing a special occurrence of an atomic variable $\bullet$ (called* ***hole****) and we denote by $C[A]$ the formula obtained by replacing $\bullet$ with a formula A. An* ***atom*** *is either the* ***unit*** *$\circ$, or a predicate $\langle x!y\rangle$ or $(x?y)$. The* ***(linear) implication*** *$A \multimap B$ is defined as $A^{\perp} \parr B$, where the* ***negation*** *is defined over formulas by extending the negation on atoms via the* ***de Morgan laws*** *in Figure 5.*

For each formula, we define the set $\mathsf{free}(A)$ of ***free variables*** *as the set of variables occurring in A which are not bounded by any quantifier. The free variables in a sequent $\Gamma = A_1, \ldots, A_n$ is the set $\mathsf{free}(\Gamma) = \bigcap_{i=1}^{n} \mathsf{free}(A_i)$. From now on, we assume sequents to be* ***clean****, that is, such that each variable x can occur bound by at most a universal quantifier, or by two dual nominal quantifiers.*

► Remark 3. To provide a lighter presentation of our systems, as well as to highlight the connections with the π-calculus, in this paper we consider formulas whose propositional atoms are generated by a limited signature containing no function symbols and two "dual" binary predicates $\langle -!-\rangle$ and $(-?-)$. However, a more expressive extension could be easily defined, and the results presented in this paper could be straightforwardly extended by addressing simple technical nuances, which we highlight in this paper whenever relevant.

► **Definition 4.** *A* ***nominal variable*** *is an element of the form x^{∇} with $x \in \mathcal{V}$ and $\nabla \in \{\text{И}, \text{Я}\}$. If $\mathcal{S}$ is a set of nominal variables, we say that x* ***occurs*** *in $\mathcal{S}$ if $x^{\text{И}}$ or $x^{\text{Я}}$ is an element of $\mathcal{S}$. A* ***(nominal) store*** *$\mathcal{S}$ is a set of nominal variables such that each variable occurs at most once in $\mathcal{S}$. A* ***judgement*** *$\mathcal{S} \vdash \Gamma$ consists of a store $\mathcal{S}$, and a clean sequent Γ.*

► **Notation 5.** *We write judgements $\mathcal{S} \vdash \Gamma$ with $\mathcal{S} = \varnothing$ (resp. $\mathcal{S} = \{x_1^{\nabla_1}, \ldots, x_n^{\nabla_n}\}$) simply as $\mathcal{S} \vdash \Gamma$ (resp. $x_1^{\nabla_1}, \ldots, x_n^{\nabla_n} \vdash \Gamma$, i.e., omitting parenthesis). We write $\mathcal{S}_1, \mathcal{S}_2$ to denote the (disjoint) union of two stores such that a same variable does not occur in both $\mathcal{S}_1$ and $\mathcal{S}_2$.*

We define rule systems using rules from Figure 6. The rules in Figure 6 in the first block are standard rules for the first-order multiplicative and additive fragment of linear logic decorated with stores. As expected, rules $\otimes$ and mix split the context (and thus the store) among premises to enforce the linear use of resources, which is typical for multiplicative rules. In contrast, the rule & (with) duplicates the context (and thus the store). The rules for the connective $\blacktriangleleft$ in the second block are also multiplicative in this sense, as they maintain the same context-splitting behavior. The rules $\text{И}_\circ$ and $\text{Я}_\circ$ simply remove quantification respecting the freshness condition ($\dagger$), as the standard rule $\forall$ for the universal quantifier. We are not making use of substitution for these rules because we assume α-renaming could be applied to the formula prior to the application of the rule, in order to satisfy the side condition $\dagger$. Similarly, the rule И_{load} (resp. Я_{load}) removes quantification respecting the freshness condition $\dagger$, as the standard rule $\forall$ for the universal quantifier, but it also adds to the store the nominal variable $x^{\text{И}}$ (resp. $x^{\text{Я}}$), where x is the variable bound by the nominal

$$\mathsf{ax}\frac{}{\mathcal{S}\vdash\langle x!y\rangle,(x?y)}\quad \parr\frac{\mathcal{S}\vdash\Gamma,A,B}{\mathcal{S}\vdash\Gamma,A\parr B}\quad \otimes\frac{\mathcal{S}_1\vdash\Gamma,A\quad \mathcal{S}_2\vdash B,\Delta}{\mathcal{S}_1,\mathcal{S}_2\vdash\Gamma,A\otimes B,\Delta}\quad \circ\frac{}{\mathcal{S}\vdash\circ}\quad \mathsf{mix}\frac{\mathcal{S}_1\vdash\Gamma\quad \mathcal{S}_2\vdash\Delta}{\mathcal{S}_1,\mathcal{S}_2\vdash\Gamma,\Delta}$$

$$\oplus\frac{\mathcal{S}\vdash\Gamma,A_i}{\mathcal{S}\vdash\Gamma,A_1\oplus A_2}\quad \&\frac{\mathcal{S}\vdash\Gamma,A\quad \mathcal{S}\vdash\Gamma,B}{\mathcal{S}\vdash\Gamma,A\& B}\quad \forall\frac{\mathcal{S}\vdash\Gamma,A}{\mathcal{S}\vdash\Gamma,\forall x.A}\dagger\quad \exists\frac{\mathcal{S}\vdash\Gamma,A[y/x]}{\mathcal{S}\vdash\Gamma,\exists x.A}$$

$$\blacktriangleleft\frac{\mathcal{S}_1\vdash\Gamma,A,C\quad \mathcal{S}_2\vdash\Delta,B,D}{\mathcal{S}_1,\mathcal{S}_2\vdash\Gamma,\Delta,A\blacktriangleleft B,C\blacktriangleleft D}\quad \blacktriangleleft_\circ\frac{\mathcal{S}_1\vdash\Gamma,A\quad \mathcal{S}_2\vdash\Delta,B}{\mathcal{S}_1,\mathcal{S}_2\vdash\Gamma,\Delta,A\blacktriangleleft B}$$

$$\text{И}_\circ\frac{\mathcal{S}\vdash\Gamma,A}{\mathcal{S}\vdash\Gamma,\text{И}x.A}\dagger\quad \text{И}_{\mathsf{load}}\frac{\mathcal{S},x^{\text{И}}\vdash\Gamma,A}{\mathcal{S}\vdash\Gamma,\text{И}x.A}\dagger\quad \text{И}_{\mathsf{pop}}\frac{\mathcal{S}\vdash\Gamma,A[y/x]}{\mathcal{S},y^{\text{И}}\vdash\Gamma,\text{Я}x.A}$$

$$\text{Я}_\circ\frac{\mathcal{S}\vdash\Gamma,A}{\mathcal{S}\vdash\Gamma,\text{Я}x.A}\dagger\quad \text{Я}_{\mathsf{load}}\frac{\mathcal{S},x^{\text{Я}}\vdash\Gamma,A}{\mathcal{S}\vdash\Gamma,\text{Я}x.A}\dagger\quad \text{Я}_{\mathsf{pop}}\frac{\mathcal{S}\vdash\Gamma,A[y/x]}{\mathcal{S},y^{\text{Я}}\vdash\Gamma,\text{И}x.A}$$

$$\mathsf{AX}\frac{}{\varnothing\vdash A,A^\perp}\quad \mathsf{cut}\frac{\mathcal{S}_1\vdash\Gamma,A\quad \mathcal{S}_2\vdash A^\perp,\Delta}{\mathcal{S}_1,\mathcal{S}_2\vdash\Gamma,\Delta}\quad \text{И-Я}\frac{\mathcal{S}\vdash\Gamma,A,B}{\mathcal{S}\vdash\Gamma,\text{Я}x.A,\text{И}x.B}\dagger$$

Figure 6 Sequent calculus rules with side conditions $\dagger := x \notin \mathsf{free}(\Gamma)$. The system PiL is made of the rules above the double line. The rules below it are derivable.

quantifier of the principal formula. The rule И_{pop} (resp. Я_{pop}) behaves similarly to the rule $\exists$ for the existential quantifier, but removing an occurrence of the dual nominal quantifier Я (resp. И). The name is due to the fact that the variable used for the substitution has to be a nominal variable $x^{\text{И}}$ (resp. $x^{\text{Я}}$) in the store.

We prove in this section the admissibility of the that the rules below the double line, which are the standard rules for the general (non-atomic) axiom and cut, and a special rule И-Я removing a pair of dual nominal quantifiers binding the same variable x.

▶ **Definition 6.** *We define the following systems using rules from Figure 6.*

$$\begin{array}{rclrcl}
\mathsf{MLL} &=& \{\mathsf{ax},\parr,\otimes\} & \mathsf{MLL}^\circ &=& \mathsf{MLL}\cup\{\circ,\mathsf{mix}\}\\
\mathsf{MALL} &=& \mathsf{MLL}\cup\{\oplus,\&\} & \mathsf{MALL}_1 &=& \mathsf{MALL}\cup\{\forall,\exists\}\\
\mathsf{NML} &=& \mathsf{MLL}\cup\{\blacktriangleleft,\blacktriangleleft_\circ,\circ,\mathsf{mix}\} & \mathsf{NMAL} &=& \mathsf{MALL}\cup\{\blacktriangleleft,\blacktriangleleft_\circ,\circ,\mathsf{mix}\}\\
\mathsf{mini\text{-}PiL} &=& \mathsf{NMAL}\cup\{\exists,\forall,\text{И}_\circ,\text{Я}_\circ,\text{И-Я}\} & \mathsf{PiL}^- &=& \mathsf{NMAL}\cup\left\{\exists,\forall,\text{И}_\circ,\text{Я}_\circ,\text{И}_{\mathsf{load}},\text{И}_{\mathsf{pop}}\right\}
\end{array}$$
$$\mathsf{PiL} = \mathsf{MALL}_1\cup\{\blacktriangleleft,\blacktriangleleft_\circ,\circ,\mathsf{mix}\}\cup\left\{\text{И}_\circ,\text{Я}_\circ,\text{И}_{\mathsf{load}},\text{И}_{\mathsf{pop}},\text{Я}_{\mathsf{load}},\text{Я}_{\mathsf{pop}}\right\} \tag{4}$$

If X *is a system, we write* $\vdash_{\mathsf{X}}\Gamma$ *to denote that* $\varnothing\vdash\Gamma$ *is derivable in* X*.*

▶ Remark 7. During proof search, the nominal quantifier $\nabla\in\{\text{И},\text{Я}\}$ could be removed by a rule ∇_{load}, storing the variable bound by the nominal quantifier in the store (as a nominal variable). Since the axiom rule and the unit rule have empty store, each nominal variable x^∇ in the store must be used for the substitution of a variable bound by the dual nominal quantifier $\nabla^\perp$. That is, any derivation establishes some pairing between each ∇_{load} with some ∇_{pop} rules above it in a derivation – in absence of additive connectives, such a ∇_{pop} above the ∇_{load} is unique.

A similar pairing can be observed in the weaker rule И-Я, which removes a pair of dual nominal quantifiers binding the same variable x. However, such a localized interaction rules out the possibility to prove quantifier swaps for nominal quantifiers and the nominal-choice laws from Equation (5).

▶ Remark 8. The pair $\langle И, Я \rangle$ of nominal quantifiers in PiL behaves differently from the pair $\langle И, Э \rangle$ considered by Horne and Tiu for the logic BV^1 and its extensions [45, 46, 48]. One difference is the way И and Я interact in PiL, in which each nominal quantifier $\eth$ interacts with at most one dual quantifier $\eth^\perp$, while in BV^1 a И can interact with multiple Э. By means of example, the implication $(Иx.A \otimes Иx.B) \multimap Иx.(A \parr B)$ (i.e., the formula $Яx.A^\perp \parr Яx.B^\perp \parr Иx.(A \parr B)$) is provable in BV^1 but not in PiL.

This reminds the different ways the modalities in the modal logics M and K interacts via the rules: in the former, each diamond ($\Diamond$)interacts with exectly one box ($\Box$), while in the latter, multiple diamonds can interact with a single box, as shown in the sequent rules of their sequent calculi – see [57, 59, 11] for additional details.

$$\mathsf{M}\,\frac{\mathcal{S} \vdash B, A}{\mathcal{S} \vdash \Diamond B, \Box A} \qquad \text{and} \qquad \mathsf{K}\,\frac{\mathcal{S} \vdash B_1, \ldots, B_n, A}{\mathcal{S} \vdash \Diamond B_1, \ldots, \Diamond B_n, \Box A}\; n \in \mathbb{N}$$

Moreover, the implication $Иx.A \multimap (Иx.A^\perp)^\perp$ (that is, the formula $Яx.A \parr Яx.A$) is not derivable in PiL, while it is in BV^1.

Another difference depends on the way nominal quantifiers interact with the connective modeling sequentiality. Our nominal quantifiers do not satisfy scope extrusion over sequentiality, that is, the formula $Иx.(A \triangleleft B) \multimapboth (Иx.A) \triangleleft B$ with $x \notin \mathsf{free}(B)$ is not derivable in PiL. However, this property is striclty needed in BV^1 in order to guarantee that the logical implication is a transitive relation (i.e., that if $A \multimap B$ and $B \multimap C$ are derivable, then also $A \multimap C$ is derivable).

4 Proof theoretical properties of PiL

In this section we prove the proof theoretical properties of the system PiL, including the derivability of rules and the possibility of embedding PiL in MAV^1.

Our systems satisfy the property referred to as *initial coherence* [15, 66], that is, the property that atomic axioms suffice to guarantee the possibility of deriving the general axiom rule. Said differently, in PiL we can derive any formula of the form $A \multimap A$ using axiom rules restricted on atoms only.

▶ **Proposition 9.** *Then the rule* И-Я *is derivable in* $\{И_{\mathsf{load}}, И_{\mathsf{pop}}\}$.

Proof. It suffices to remark that each instance of И-Я can be replaced by a $И_{\mathsf{load}}$ followed (bottom-up) by a $И_{\mathsf{pop}}$. ◀

▶ **Proposition 10.** *Then the rule* AX *is derivable in* $\mathsf{MALL}^1 \cup \{\circ, \triangleleft, И_{\mathsf{load}}, И_{\mathsf{pop}}\}$.

Proof. We that the judgement $\mathcal{S} \vdash A^\perp, A$ is derivable in $\mathsf{MALL}^1 \cup \{\circ, \triangleleft, И\text{-}Я\}$ for any formula A by induction on the structure of A:

- if $A = \circ$, then such a derivation is made of a mix with premises of the form $\mathcal{S} \vdash \circ$, each conclusion of a $\circ$-rule;
- if $A = B \triangleleft C$, then $A^\perp = B^\perp \triangleleft C^\perp$. We apply a rule $\triangleleft$, and we conclude by inductive hypothesis;
- if $A = \eth x.B$ with $\nabla \in \{И, Я\}$, then we apply (bottom-up) a rule И-Я-rule and we conclude by inductive hypothesis;
- otherwise, we proceed as standard in MALL_1.

The statement follow from Proposition 9. ◀

We have the following properties for our connectives, quantifiers and unit. Note that the list in Equation (5) is not complete, since additional implications and equivalences immediately follow by duality.

▶ **Proposition 11.** *The following logical equivalences and implications are derivable in* PiL*.*

Unit Laws	*Monoidal Laws*	
$(A \parr \circ) \multimapboth (A \otimes \circ) \multimapboth A$ $(A \blacktriangleleft \circ) \multimapboth (\circ \blacktriangleleft A) \multimapboth A$ $(\circ \,\&\, \circ) \multimapboth (\circ \oplus \circ) \multimapboth \circ$ $\forall x.\circ \multimapboth \text{И}x.\circ \multimapboth \text{Я}x.\circ \multimapboth \exists x.\circ \multimapboth \circ$	$A \odot B \multimapboth B \odot A$ $(A \odot B) \odot C \multimapboth A \odot (B \odot C)$ *with* $\odot \in \{\parr, \otimes, \oplus, \&\}$	
Scope extrusion	*Quantifier Swap*	
$\text{И}x.(A \parr B) \multimapboth (\text{И}x.A) \parr B$ $\text{И}x.B \multimapboth B$ *if* $x \notin \mathsf{free}(B)$	$\mho x.\mho y.A \multimapboth \mho y.\mho x.A$ *with* $\mho \in \{\exists, \text{Я}, \text{И}, \forall\}$	(5)
Multiplicative refinement	*Quantifier refinement*	
$(A \otimes B) \multimap (A \blacktriangleleft B)$ $(A \blacktriangleleft B) \multimap (A \parr B)$	$\forall x.A \multimap \text{И}x.A \qquad \text{И}x.A \multimap \exists x.A$ $\forall x.A \multimap \text{Я}x.A \qquad \text{Я}x.A \multimap \exists x.A$	
Nominal-choice	*Distributivity of choice*	
$(\text{И}x.A \,\&\, \text{И}x.B) \multimapboth (\text{И}x.(A \,\&\, B))$ $(\text{И}x.A \oplus \text{И}x.B) \multimapboth (\text{И}x.(A \oplus B))$	$((A \parr B) \,\&\, (A \parr C)) \multimap (A \parr (B \,\&\, C))$ $(A \parr (B \,\&\, C)) \multimap ((A \parr B) \,\&\, (A \parr C))$ $(A \parr (B \oplus C)) \multimap ((A \oplus B) \parr (A \oplus C))$	

Proof. Unit laws follows by the existence of the following derivations.

$$\parr\dfrac{\otimes\dfrac{\mathsf{AX}\dfrac{}{\vdash A^\perp, A} \quad \circ\dfrac{}{\vdash \circ}}{\vdash (A^\perp \otimes \circ), A}}{\vdash (A^\perp \otimes \circ) \parr A} \qquad \parr{\times}2\dfrac{\mathsf{mix}\dfrac{\mathsf{AX}\dfrac{}{\vdash A^\perp, A} \quad \circ\dfrac{}{\vdash \circ}}{\vdash A^\perp, A, \circ}}{\vdash A^\perp \parr (A \parr \circ)} \qquad \parr\dfrac{\blacktriangleleft_\circ\dfrac{\mathsf{AX}\dfrac{}{\vdash A^\perp, A} \quad \circ\dfrac{}{\vdash \circ}}{\vdash A^\perp \blacktriangleleft \circ, A}}{\vdash (A^\perp \blacktriangleleft \circ) \parr A}$$

$$\parr\dfrac{\oplus\dfrac{\mathsf{AX}\dfrac{}{\vdash \circ, \circ}}{\vdash \circ \oplus \circ, \circ}}{\vdash (\circ \oplus \circ) \parr \circ} \qquad \parr\dfrac{\&\dfrac{\mathsf{AX}\dfrac{}{\vdash \circ, \circ} \quad \mathsf{AX}\dfrac{}{\vdash \circ, \circ}}{\vdash \circ \,\&\, \circ, \circ}}{\vdash (\circ \,\&\, \circ) \parr \circ} \qquad \otimes\dfrac{\parr\dfrac{\mho\dfrac{\mathsf{AX}\dfrac{}{\vdash \circ, \circ}}{\vdash \circ, \mho x.\circ}}{\vdash \circ \parr \mho x.\circ} \quad \parr\dfrac{\mho^\perp\dfrac{\mathsf{AX}\dfrac{}{\vdash \circ, \circ}}{\vdash \mho^\perp x.\circ, \circ}}{\vdash \mho^\perp x.\circ \parr \circ}}{\vdash \mho x.\circ \multimapboth \circ}$$

Monodail laws are proven as standard in MALL. Scope extrusion and nominal quantifiers swaps are proven as shown in Equation (6) below.

$$2{\times}\parr\dfrac{\text{И}_{\mathsf{load}}\dfrac{\text{И}_{\mathsf{pop}}\dfrac{\otimes\dfrac{\mathsf{AX}\dfrac{}{\vdash A^\perp, A} \quad \mathsf{AX}\dfrac{}{\vdash B^\perp, B}}{\vdash (A^\perp \otimes B^\perp), A, B}}{x^{\text{И}} \vdash \text{Я}x.(A^\perp \otimes B^\perp), A, B}}{\vdash \text{Я}x.(A^\perp \otimes B^\perp), \text{И}x.A, B}}{\vdash \text{Я}x.(A^\perp \otimes B^\perp) \parr ((\text{И}x.A) \parr B)} \qquad 2{\times}\text{И}_{\mathsf{load}}\dfrac{2{\times}\text{И}_{\mathsf{pop}}\dfrac{\mathsf{AX}\dfrac{}{\vdash A^\perp, A}}{x^{\text{И}}, y^{\text{И}} \vdash \text{Я}y.\text{Я}x.A^\perp}}{\vdash \text{И}x.\text{И}y.A \parr \text{Я}y.\text{Я}x.A^\perp} \tag{6}$$

Quantifier swap for universal and existential quantifiers are standard as in in MALL_1. For multiplicative refinement we only show the derivation for $(A \otimes B) \multimap (A \blacktriangleleft B)$ on the left of

Equation (7), since the derivation for $(A \blacktriangleleft B) \multimap (A \parr B)$ is similar. Nominal refinements are proven as shown in the right of Equation (7) below.

$$
{\scriptstyle\parr}\dfrac{{\scriptstyle\parr}\dfrac{{\scriptstyle\blacktriangleleft_\circ}\dfrac{{\scriptstyle\mathsf{AX}}\dfrac{}{\vdash A, A^\perp} \qquad {\scriptstyle\mathsf{AX}}\dfrac{}{\vdash B, B^\perp}}{\vdash A^\perp, B^\perp, A \blacktriangleleft B}}{\vdash A^\perp \parr B^\perp, A \blacktriangleleft B}}{\vdash (A^\perp \parr B^\perp) \parr (A \blacktriangleleft B)}
\qquad\qquad
{\scriptstyle\parr}\dfrac{{\scriptstyle\nabla_\circ}\dfrac{{\scriptstyle\exists}\dfrac{{\scriptstyle\mathsf{AX}}\dfrac{}{\vdash A^\perp, A}}{\vdash \exists x.A^\perp, A}}{\vdash \exists x.A^\perp, \nabla x.A}}{\vdash \exists x.A^\perp \parr \nabla x.A}
\tag{7}
$$

Finally, distributivity of the choice are standard in MALL (they are proven by applying (bottom-up) $\parr$ and $\&$ rules first, followed by $\oplus$ and $\otimes$ rules.), and nominal-choice laws are proven as follows.

$$
\begin{gathered}
{\scriptstyle\otimes}\dfrac{
{\scriptstyle\parr}\dfrac{{\scriptstyle\&}\dfrac{
{\scriptstyle\text{И}_{\mathsf{load}}}\dfrac{{\scriptstyle\text{И}_{\mathsf{pop}}}\dfrac{{\scriptstyle\oplus}\dfrac{{\scriptstyle\mathsf{AX}}\dfrac{}{\vdash A^\perp, A}}{\vdash A^\perp \oplus B^\perp, A}}{x^{\text{И}} \vdash \text{Я}x.(A^\perp \oplus B^\perp), A}}{\vdash \text{Я}x.(A^\perp \oplus B^\perp), \text{И}x.A}
\quad
{\scriptstyle\text{И}_{\mathsf{load}}}\dfrac{{\scriptstyle\text{И}_{\mathsf{pop}}}\dfrac{{\scriptstyle\oplus}\dfrac{{\scriptstyle\mathsf{AX}}\dfrac{}{\vdash B^\perp, B}}{\vdash A^\perp \oplus B^\perp, B}}{x^{\text{И}} \vdash \text{Я}x.(A^\perp \oplus B^\perp), B}}{\vdash \text{Я}x.(A^\perp \oplus B^\perp), \text{И}x.B}
}{\vdash \text{Я}x.(A^\perp \oplus B^\perp), \text{И}x.A \,\&\, \text{И}x.B}}{\vdash \text{Я}x.(A^\perp \oplus B^\perp) \parr (\text{И}x.A \,\&\, \text{И}x.B)}
\quad
{\scriptstyle\parr}\dfrac{{\scriptstyle\text{И}_{\mathsf{load}}}\dfrac{{\scriptstyle\&}\dfrac{
{\scriptstyle\oplus}\dfrac{{\scriptstyle\text{И}_{\mathsf{pop}}}\dfrac{{\scriptstyle\mathsf{AX}}\dfrac{}{\vdash A^\perp, A}}{x^{\text{И}} \vdash \text{Я}x.A^\perp, A}}{x^{\text{И}} \vdash \text{Я}x.A^\perp \oplus \text{Я}x.B^\perp, A}
\quad
{\scriptstyle\oplus}\dfrac{{\scriptstyle\text{И}_{\mathsf{pop}}}\dfrac{{\scriptstyle\mathsf{AX}}\dfrac{}{\vdash B^\perp, B}}{x^{\text{И}} \vdash \text{Я}x.B^\perp, B}}{x^{\text{И}} \vdash \text{Я}x.A^\perp \oplus \text{Я}x.B^\perp, B}
}{x^{\text{И}} \vdash \text{Я}x.A^\perp \oplus \text{Я}x.B^\perp, A \,\&\, B}}{\vdash \text{Я}x.A^\perp \oplus \text{Я}x.B^\perp, \text{И}x.(A \,\&\, B)}}{\vdash (\text{Я}x.A^\perp \oplus \text{Я}x.B^\perp) \parr \text{И}x.(A \,\&\, B)}
}{\vdash (\text{И}x.A \,\&\, \text{И}x.B) \multimapboth (\text{И}x.(A \,\&\, B))}
\\[2ex]
{\scriptstyle\otimes}\dfrac{
{\scriptstyle\parr}\dfrac{{\scriptstyle\&}\dfrac{
{\scriptstyle\text{И}_{\mathsf{load}}}\dfrac{{\scriptstyle\text{И}_{\mathsf{pop}}}\dfrac{{\scriptstyle\oplus}\dfrac{{\scriptstyle\mathsf{AX}}\dfrac{}{\vdash A^\perp, A}}{\vdash A^\perp, A \oplus B}}{x^{\text{И}} \vdash \text{Я}x.A^\perp, A \oplus B}}{\vdash \text{Я}x.A^\perp, \text{И}x.(A \oplus B)}
\quad
{\scriptstyle\text{И}_{\mathsf{load}}}\dfrac{{\scriptstyle\text{И}_{\mathsf{pop}}}\dfrac{{\scriptstyle\oplus}\dfrac{{\scriptstyle\mathsf{AX}}\dfrac{}{\vdash B^\perp, B}}{\vdash B^\perp, A \oplus B}}{x^{\text{И}} \vdash \text{Я}x.B^\perp, A \oplus B}}{\vdash \text{Я}x.B^\perp, \text{И}x.(A \oplus B)}
}{\vdash \text{Я}x.A^\perp \,\&\, \text{Я}x.B^\perp, \text{И}x.(A \oplus B)}}{\vdash (\text{Я}x.A^\perp \,\&\, \text{Я}x.B^\perp) \parr \text{И}x.(A \oplus B)}
\quad
{\scriptstyle\parr}\dfrac{{\scriptstyle\text{Я}_{\mathsf{load}}}\dfrac{{\scriptstyle\&}\dfrac{
{\scriptstyle\text{Я}_{\mathsf{pop}}}\dfrac{{\scriptstyle\text{И}_{\mathsf{pop}}}\dfrac{{\scriptstyle\mathsf{AX}}\dfrac{}{\vdash A, A^\perp}}{\vdash A, A^\perp \oplus B^\perp}}{x^{\text{Я}} \vdash \text{И}x.A, A^\perp \oplus B^\perp}
\quad
{\scriptstyle\text{Я}_{\mathsf{pop}}}\dfrac{{\scriptstyle\text{И}_{\mathsf{pop}}}\dfrac{{\scriptstyle\mathsf{AX}}\dfrac{}{\vdash B, B^\perp}}{\vdash B, A^\perp \oplus B^\perp}}{x^{\text{Я}} \vdash \text{И}x.A, A^\perp \oplus B^\perp}
}{x^{\text{Я}} \vdash \text{И}x.A \,\&\, \text{И}x.B, A^\perp \oplus B^\perp}}{\vdash \text{И}x.A \,\&\, \text{И}x.B, \text{Я}x.(A^\perp \oplus B^\perp)}}{\vdash (\text{И}x.A \,\&\, \text{И}x.B) \parr \text{Я}x.(A^\perp \oplus B^\perp)}
}{\vdash (\text{И}x.A \oplus \text{И}x.B) \multimapboth (\text{И}x.(A \oplus B))}
\end{gathered}
\tag{8}
$$

◄

► Remark 12. The system mini-PiL satisfies all the equivalences and implications in Proposition 11, except for quantifier swap and nominal choices, while the system PiL^- satisfies all the equivalences and implications in Proposition 11, except for the bottom-most nominal choice. For this latter, it only satisfies the left-to-right implication $\vdash_{\mathsf{PiL}^-} (\text{И}x.A \oplus \text{И}x.B) \multimap (\text{И}x.(A \oplus B))$ (see the right branch of the bottom-most derivation in Equation (8)), but not the converse.

4.1 Cut-Elimination

We prove the admissibility of the rule cut, we provide a cut-elimination procedure adapting the one for MALL_1. In absence of the nominal quantifier, the proof taking into account the connective $\blacktriangleleft$ is straightforward. In the presence of the nominal quantifier, the proof is more intricate because of the implicit links between ∇_{load}-rules and ∇_{pop}-rules in a derivation we discuss in Remark 7. For example, consider the derivation with cut of the formula $\text{И}x.a \multimap \nabla x.a$ in the left of Equation (9), where we marked the flows of the nominal variables.

$$\text{cut}\,\dfrac{\mathcal{S} \vdash \Gamma, a \qquad \text{ax}\,\dfrac{}{\mathcal{S} \vdash a^\perp, a}}{\mathcal{S} \vdash \Gamma, a} \rightsquigarrow \mathcal{S} \vdash \Gamma, a \qquad\qquad \text{mix}\,\dfrac{\mathcal{S} \vdash \Gamma \qquad \text{cut}\,\dfrac{\circ\,\dfrac{}{\mathcal{S} \vdash \circ} \quad \circ\,\dfrac{}{\mathcal{S} \vdash \circ}}{\mathcal{S} \vdash}}{\mathcal{S} \vdash \Gamma} \rightsquigarrow \mathcal{S} \vdash \Gamma$$

$$\text{cut}\,\dfrac{\parr\,\dfrac{\mathcal{S} \vdash \Gamma, A, B}{\mathcal{S} \vdash \Gamma, A \parr B} \qquad \otimes\,\dfrac{\mathcal{S}_1 \vdash A^\perp, \Delta_1 \quad \mathcal{S}_2 \vdash B^\perp, \Delta_2}{\mathcal{S}_1, \mathcal{S}_2 \vdash A^\perp \otimes B^\perp, \Delta_1, \Delta_2}}{\mathcal{S}, \mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta_1, \Delta_2} \rightsquigarrow \text{cut}\,\dfrac{\text{cut}\,\dfrac{\mathcal{S} \vdash \Gamma, A, B \quad \mathcal{S}_1 \vdash A^\perp, \Delta_1}{\mathcal{S} \vdash \Gamma, \Delta_1, A} \qquad \mathcal{S}_2 \vdash B^\perp, \Delta_2}{\mathcal{S} \vdash \Gamma, \Delta_1, \Delta_2}$$

$$\text{cut}\,\dfrac{\oplus\,\dfrac{\mathcal{S}_1 \vdash \Gamma, A_i}{\mathcal{S}_1 \vdash \Gamma, A_1 \oplus A_2} \qquad \&\,\dfrac{\mathcal{S}_2 \vdash {A_1}^\perp, \Delta \quad \mathcal{S}_2 \vdash {A_2}^\perp, \Delta}{\mathcal{S}_2 \vdash {A_1}^\perp \,\&\, {A_2}^\perp, \Delta}}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta} \rightsquigarrow \text{cut}\,\dfrac{\mathcal{S}_1 \vdash \Gamma, A_i \quad \mathcal{S}_2 \vdash {A_i}^\perp, \Delta}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta}$$

$$\text{cut}\,\dfrac{\forall\,\dfrac{\overset{\mathcal{D}}{\mathcal{S}_1 \vdash \Gamma, A^\perp}}{\mathcal{S}_1 \vdash \Gamma, \forall x.A^\perp} \qquad \exists\,\dfrac{\mathcal{S}_2 \vdash A[c/x], \Delta}{\mathcal{S}_2 \vdash \exists x.A, \Delta}}{\mathcal{S} \vdash \Gamma, \Delta} \rightsquigarrow \text{cut}\,\dfrac{\overset{\mathcal{D}[c/x]}{\mathcal{S}_1 \vdash \Gamma, A^\perp[c/x]} \quad \mathcal{S}_2 \vdash A[c/x], \Delta}{\mathcal{S} \vdash \Gamma, \Delta}$$

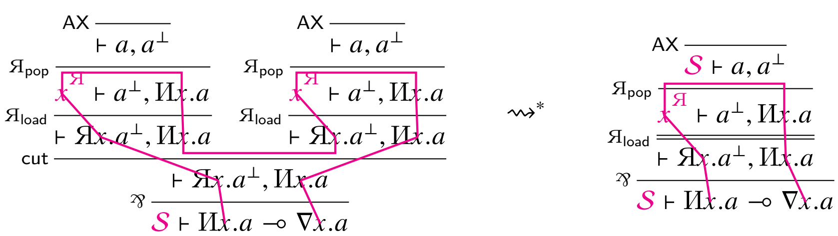
Figure 7 Cut-elimination steps for MALL_1 and MLL°, where $\mathcal{D}[c/x]$ is the derivation obtained by replacing all occurrences of x in $\mathcal{D}$ with c.

$$\text{cut}\,\dfrac{\blacktriangleleft\,\dfrac{\mathcal{S}_1 \vdash \Gamma_1, C, A \quad \mathcal{S}_2 \vdash \Gamma_2, D, B}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma_1, \Gamma_2, C \blacktriangleleft D, A \blacktriangleleft B} \quad \blacktriangleleft\,\dfrac{\mathcal{S}_3 \vdash A^\perp, E, \Delta_1 \quad \mathcal{S}_4 \vdash B^\perp, F, \Delta_2}{\mathcal{S}_3, \mathcal{S}_4 \vdash A^\perp \blacktriangleleft B^\perp, E \blacktriangleleft F, \Delta_1, \Delta_2}}{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4 \vdash \Gamma_1, \Gamma_2, C \blacktriangleleft D, E \blacktriangleleft F, \Delta_1, \Delta_2} \rightsquigarrow \blacktriangleleft\,\dfrac{\text{cut}\,\dfrac{\mathcal{S}_1 \vdash \Gamma_1, C, A \quad \mathcal{S}_3 \vdash A^\perp, E, \Delta_1}{\mathcal{S}_1, \mathcal{S}_3 \vdash \Gamma_1, C, E, \Delta_1} \quad \text{cut}\,\dfrac{\mathcal{S}_2 \vdash \Gamma_2, D, B \quad \mathcal{S}_4 \vdash B^\perp, F, \Delta_2}{\mathcal{S}_2, \mathcal{S}_4 \vdash \Gamma_2, D, F, \Delta_2}}{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4 \vdash \Gamma_1, \Gamma_2, C \blacktriangleleft D, E \blacktriangleleft F, \Delta_1, \Delta_2}$$

$$\text{cut}\,\dfrac{\blacktriangleleft\,\dfrac{\mathcal{S}_1 \vdash \Gamma_1, C, A \quad \mathcal{S}_2 \vdash \Gamma_2, D, B}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma_1, \Gamma_2, C \blacktriangleleft D, A \blacktriangleleft B} \quad \blacktriangleleft_\circ\,\dfrac{\mathcal{S}_3 \vdash A^\perp, \Delta_1 \quad \mathcal{S}_4 \vdash B^\perp, \Delta_2}{\mathcal{S}_3, \mathcal{S}_4 \vdash A^\perp \blacktriangleleft B^\perp, \Delta_1, \Delta_2}}{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4 \vdash \Gamma_1, \Gamma_2, C \blacktriangleleft D, \Delta_1, \Delta_2} \rightsquigarrow \blacktriangleleft_\circ\,\dfrac{\text{cut}\,\dfrac{\mathcal{S}_1 \vdash \Gamma_1, C, A \quad \mathcal{S}_3 \vdash A^\perp, \Delta_1}{\mathcal{S}_1, \mathcal{S}_3 \vdash \Gamma_1, C, \Delta_1} \quad \text{cut}\,\dfrac{\mathcal{S}_2 \vdash \Gamma_2, D, B \quad \mathcal{S}_4 \vdash B^\perp, \Delta_2}{\mathcal{S}_2, \mathcal{S}_4 \vdash \Gamma_2, D, \Delta_2}}{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4 \vdash \Gamma_1, \Gamma_2, C \blacktriangleleft D, \Delta_1, \Delta_2}$$

$$\text{cut}\,\dfrac{\blacktriangleleft_\circ\,\dfrac{\mathcal{S}_1 \vdash \Gamma_1, A \quad \mathcal{S}_2 \vdash \Gamma_2, B}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma_1, \Gamma_2, A \blacktriangleleft B} \quad \blacktriangleleft_\circ\,\dfrac{\mathcal{S}_3 \vdash A^\perp, \Delta_1 \quad \mathcal{S}_4 \vdash B^\perp, \Delta_2}{\mathcal{S}_3, \mathcal{S}_4 \vdash A^\perp \blacktriangleleft B^\perp, \Delta_1, \Delta_2}}{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4 \vdash \Gamma_1, \Gamma_2, \Delta_1, \Delta_2} \rightsquigarrow \text{mix}\,\dfrac{\text{cut}\,\dfrac{\mathcal{S}_1 \vdash \Gamma_1, A \quad \mathcal{S}_3 \vdash A^\perp, \Delta_1}{\mathcal{S}_1, \mathcal{S}_3 \vdash \Gamma_1, \Delta_1} \quad \text{cut}\,\dfrac{\mathcal{S}_2 \vdash \Gamma_2, B \quad \mathcal{S}_4 \vdash B^\perp, \Delta_2}{\mathcal{S}_2, \mathcal{S}_4 \vdash \Gamma_2, \Delta_1}}{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4 \vdash \Gamma_1, \Gamma_2, \Delta_1, \Delta_2}$$

Figure 8 Cut-elimination steps for the connective $\blacktriangleleft$ and its rules.



(9)

In order to perform cut-elimination, we need to be able to keeping track of the variables bound by dual nominal quantifiers, which are supposed to be linked by the cut-rule, even when the nominal quantifiers are removed. For this purpose, we introduce the following auxiliary ***store-cut*** rule we use during the rewriting process of cut-elimination.

$$\mathcal{S}\text{-cut}\,\dfrac{\mathcal{S}, x^{И}, x^{Я} \vdash \Gamma}{\mathcal{S} \vdash \Gamma} \tag{10}$$

► **Theorem 13** (Cut elimination). *Let* Γ *a non-empty sequent. If* $\vdash_{\mathsf{PiL} \cup \{\mathsf{cut}\}} \Gamma$*, then* $\vdash_{\mathsf{PiL}} \Gamma$.

Proof. We define the *weight* of a cut-rule r in a derivation $\mathcal{D}$ as a pair $\langle \mathsf{d}_{\mathcal{D}}(\mathsf{r}), \mathsf{c}_{\mathcal{D}}(\mathsf{r})\rangle$ where $\mathsf{d}_{\mathcal{D}}(\mathsf{r})$ is the maximal distance of r from a leaf above it in the derivation tree, and $\mathsf{c}_{\mathcal{D}}(\mathsf{r})$ is the complexity of the active formula(s) of r. The ***weight*** of a $\mathcal{S}$-cut-rule r is defined similarly as $\langle \mathsf{d}_{\mathcal{D}}(\mathsf{r}), 0\rangle$. The *weight* of a derivation is the multiset of the weights of its cut-rules and $\mathcal{S}$-cut-rules.

To prove cut-elimination it suffices to apply the *cut-elimination steps* in Figures 7–9 to a top-most cut-rule or $\mathcal{S}$-cut-rule in the derivation tree. The fact that we consider the procedure to operate on a derivation whose conclusion and premises judgements have empty stores and non-empty sequents ensures that the case analysis we consider covers all the possible cases. In particular, the case $\circ$-vs-$\circ$ in Figure 7 for the $\circ$ (because the sequent in the conclusion cannot be empty), and the bottom-most case in Figure 9 (because the store in the conclusion and in the premises is empty).

In order to be able to apply this strategy, as standard in the literature, we consider the *commutative cut-elimination steps*, that is, rule permutations as the ones in Figure 16 involving a cut-rule or a $\mathcal{S}$-cut-rule, allowing us to permute an instance of such rule above another rules. The termination of cut-elimination follows by the fact that each cut-elimination step applied to a top-most cut-rule r decreases $\mathsf{c}_{\mathcal{D}}(\mathsf{r})$, while each commutative cut-elimination step applied to the top-most cut-rule, or to a $\mathcal{S}$-cut-rule reduces $\mathsf{d}_{\mathcal{D}}(\mathsf{r})$. Note that a commutative step moving a cut-rule above a &-rule duplicate the cut-rule. This is why we have to define the weight as a multiset: even if the complexity does not change, the maximal distance from these two new cut-rules from the leaves is strictly smaller than the one of the original one. ◄

► Remark 14. In systems containing a self-dual unit $\circ$, which is the same unit for conjunction and disjunction, such as multiplicative linear logic with mix [33], Pomset logic [75] and BV [40], it is possible to derive the empty sequent. This depends on the fact the empty sequent is not interpreted as false (as in classical logic), but rather as the unit $\circ$, which is provable, and that the non-admissibility of the weakening rule (as in relevant logics [12, 10]) would not entail the possibility of deriving any sequent. Citing Girard (as reported in [18]) “if one were to accept this rule [mix], the good taste would require to add the void sequent as an axiom (without weakening this has no dramatic consequence)”. This explains the structure of the cut-elimination step $\circ$-vs-$\circ$ in Figure 7.

► Remark 15. If we consider a system where the only rule for nominal quantifier is the rule И-Я, then all judgements in derivations have empty store.

Note that the system $\mathsf{PiL} \setminus \{И_{\mathsf{pop}}, Я_{\mathsf{load}}\}$ presented in [60] also satisfy cut-elimination, and it already expressive enought to support the interpreting derivations as execution trees (see Section 7). Indeed, the only difference is that in such a system the nominal-choice law $(Иx.A \oplus Иx.B) \multimapboth (Иx.(A \oplus B))$ does not hold, but only the left-to-right implication is provable as explained in Remark 12.

► **Corollary 16.** *The linear implication ($\multimap$) in* PiL *defines a transitive relation, that is, if* $\vdash_{\mathsf{PiL}} A \multimap B$ *and* $\vdash_{\mathsf{PiL}} B \multimap C$*, then* $\vdash_{\mathsf{PiL}} A \multimap C$*.*

Proof. If $\vdash_{\mathsf{PiL}} A \multimap B$, then there is a derivation $\mathcal{D}^{-}_{A\multimap B}$ with conclusion $\mathcal{S} \vdash A^{\perp}, B$ because the rule $\parr$ is invertible (that is, its conclusion is derivable iff its premise is so). For the same reason, by hypothesis, there is a derivation $\mathcal{D}^{-}_{B\multimap C}$ in PiL with conclusion $\mathcal{S} \vdash B^{\perp}, C$. Thus a derivation with conclusion $\mathcal{S} \vdash A \multimap C$ made of (bottom-up) a $\parr$-rule followed by a cut-rule whose premises are the conclusion of $\mathcal{D}^{-}_{A\multimap B}$ and $\mathcal{D}^{-}_{B\multimap C}$. We conclude by applying cut-elimination. ◄

$$\mathsf{cut}\,\dfrac{\nabla_{\mathsf{load}}\dfrac{\mathcal{S}_1, x^{\nabla} \vdash \Gamma, A}{\mathcal{S}_1 \vdash \Gamma, \nabla x.A} \qquad \nabla_{\mathsf{pop}}\dfrac{\mathcal{S}_2 \vdash A^{\perp}, \Delta}{\mathcal{S}_2, x^{\nabla} \vdash \nabla^{\perp} x.A^{\perp}, \Delta}}{\mathcal{S}_1, \mathcal{S}_2, x^{\nabla} \vdash \Gamma, \Delta} \rightsquigarrow \mathsf{cut}\,\dfrac{\mathcal{S}_1, x^{\nabla} \vdash \Gamma, A^{\perp} \qquad \mathcal{S}_2 \vdash A^{\perp}, \Delta}{\mathcal{S}_1, \mathcal{S}_2, x^{\nabla} \vdash \Gamma, \Delta}$$

$$\mathsf{cut}\,\dfrac{\nabla_{\mathsf{load}}\dfrac{\begin{matrix}\Vert\,\mathcal{D}[x^{\nabla}]\\ \mathcal{S}_1, x^{\nabla} \vdash \Gamma, A\end{matrix}}{\mathcal{S}_1 \vdash \Gamma, \nabla x.A} \qquad \nabla^{\perp}_{\circ}\dfrac{\mathcal{S}_2 \vdash A^{\perp}, \Delta}{\mathcal{S}_2 \vdash \nabla^{\perp} x.A^{\perp}, \Delta}}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta} \rightsquigarrow \mathsf{cut}\,\dfrac{\begin{matrix}\Vert\,\mathcal{D}[\varnothing{\uparrow}x^{\nabla}]\\ \mathcal{S}_1 \vdash \Gamma, A\end{matrix} \qquad \mathcal{S}_2 \vdash A^{\perp}, \Delta}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta}$$

$$\nabla_{\mathsf{load}}\dfrac{\begin{matrix}\mathsf{cut}\,\dfrac{\nabla^{\perp}_{\mathsf{pop}}\dfrac{\mathcal{S}_1 \vdash \Gamma, A}{\mathcal{S}_1, x^{\nabla} \vdash \Gamma, \nabla^{\perp} x.A} \qquad \nabla_{\circ}\dfrac{\mathcal{S}_2 \vdash A^{\perp}, \Delta}{\mathcal{S}_2 \vdash \nabla x.A^{\perp}, \Delta}}{\mathcal{S}_1, \mathcal{S}_2, x^{\nabla} \vdash \Gamma, \Delta}\\ \Vert\,\mathcal{D}[x^{\nabla}]\\ \mathcal{S}_1, \mathcal{S}_2, x^{\nabla} \vdash \Gamma', B\end{matrix}}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma', \nabla x.B} \rightsquigarrow \nabla_{\circ}\dfrac{\begin{matrix}\mathsf{cut}\,\dfrac{\mathcal{S}_1 \vdash \Gamma, A \qquad \mathcal{S}_2 \vdash A^{\perp}, \Delta}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta}\\ \Vert\,\mathcal{D}[\varnothing/x^{\nabla}]\\ \mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma', B\end{matrix}}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma', \nabla y.B}$$

$$\mathsf{cut}\,\dfrac{\text{И}_{\circ}\dfrac{\mathcal{S}_1 \vdash \Gamma, A}{\mathcal{S}_1 \vdash \Gamma, \text{И}x.A} \qquad \text{Я}_{\circ}\dfrac{\mathcal{S}_2 \vdash A^{\perp}, \Delta}{\mathcal{S}_2 \vdash \text{Я}x.A^{\perp}, \Delta}}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta} \rightsquigarrow \mathsf{cut}\,\dfrac{\mathcal{S}_1 \vdash \Gamma, A \qquad \mathcal{S}_2 \vdash \Delta, A^{\perp}}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta}$$

$$\mathsf{cut}\,\dfrac{\text{И}_{\mathsf{load}}\dfrac{\mathcal{S}_1, x^{\text{И}} \vdash \Gamma, A}{\mathcal{S}_1 \vdash \Gamma, \text{И}x.A} \qquad \text{Я}_{\mathsf{load}}\dfrac{\mathcal{S}_2, x^{\text{Я}} \vdash A^{\perp}, \Delta}{\mathcal{S}_2 \vdash \text{Я}x.A^{\perp}, \Delta}}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta} \rightsquigarrow \mathcal{S}\text{-}\mathsf{cut}\,\dfrac{\mathsf{cut}\,\dfrac{\mathcal{S}_1, x^{\text{И}} \vdash \Gamma, A \qquad \mathcal{S}_2, x^{\text{Я}} \vdash A^{\perp}, \Delta}{\mathcal{S}_1, \mathcal{S}_2, x^{\text{И}}, x^{\text{Я}} \vdash \Gamma, \Delta}}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta}$$

$$\mathcal{S}\text{-}\mathsf{cut}\,\dfrac{\mathsf{cut}\,\dfrac{\text{И}_{\mathsf{pop}}\dfrac{\mathcal{S}_1 \vdash \Gamma, A}{\mathcal{S}_1, x^{\text{Я}} \vdash \Gamma, \text{И}x.A} \qquad \text{Я}_{\mathsf{pop}}\dfrac{\mathcal{S}_2 \vdash A^{\perp}, \Delta}{\mathcal{S}_2, x^{\text{И}} \vdash \text{Я}x.A^{\perp}, \Delta}}{\mathcal{S}_1, \mathcal{S}_2, x^{\text{Я}}, x^{\text{И}} \vdash \Gamma, \Delta}}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta} \rightsquigarrow \mathsf{cut}\,\dfrac{\mathcal{S}_1 \vdash \Gamma, A \qquad \mathcal{S}_2 \vdash A^{\perp}, \Delta}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta}$$

Figure 9 Cut-elimination steps for the nominal quantifiers where $\mathcal{D}[\varnothing/x^{\nabla}]$ is the derivation obtained by removing all occurrences of x^{∇} in $\mathcal{D}$, and $\mathcal{D}[\varnothing \uparrow x^{\nabla}]$ is the derivation obtained by removing all occurrences of x^{∇} in $\mathcal{D}$, and replacing any rule ∇_{pop} introducing x^{∇} in the store with a rule $\nabla_{\circ}$.

We conclude this section stating that PiL can be embedded in MAV^1 [48] using a translation $\lceil\cdot\rceil$ replacing each occurrence of $\blacktriangleleft$ with a $\triangleleft$, and each occurrence of Я with a Э. Formal definitions and details of the proof are provided in Appendix A.

▶ **Theorem 17.** *Let $A_1, \ldots, A_n$ be formulas. If $\vdash_{\mathsf{PiL}} A_1, \ldots, A_n$, then $\vdash_{\mathsf{MAV}^1} \bigparr_{i=1}^{n} \lceil A_i \rceil$.*

5 Proof Nets for PiL

In this section, we define proof nets for PiL and we prove soundness and completeness of this syntax by providing sequentialization and proof translation (desequentialization) procedures.

To handle the interaction between multiplicative ($\parr$, $\otimes$ and $\blacktriangleleft$) and additive ($\oplus$ and $\&$) connectives in PiL in a canonical way, we develop two syntaxes for proof nets:

- *conflict nets* (extending [53, 52]): these proof nets provide canonical representative for proofs modulo local rule permutations, with polynomial-time correctness criterion, sequentialization, and proof translation; and
- *slice nets* [56] (extending [55, 56]): these proof nets provide canonical representative for proofs modulo rule permutations (inlcuding non-local ones), with polynomial-time correcness criterion and sequentialization, but no polynomial time proof translation.

In particular, the former syntax is optimal for the application we aim at in this paper (see Section 6).

In the reminder of this section, we first provide some shared definitions on links and dualizers. We then define conflict nets, proving soundness and completeness by providing procedures for sequentialization and proof translation. We then define slice nets and we prove soundness and completeness using the results on conflict nets.

We first adapt the definitions of links to take into account stores.

▶ **Definition 18.** *A* ***link*** a *on a judgement* $\mathcal{S} \vdash \Gamma$ *(or a sequent* Γ*) is either*

- *a* ***sub-judgement*** *of* $\mathcal{S} \vdash \Gamma$*, that is, a judgement* $\mathcal{S}' \vdash \Gamma'$ *such that* Γ' *is an induced sub-forest of* Γ *and* $\mathcal{S}' \subseteq S$*; or*
- ***nominal link****: a pair* $\{x, y\}$ *of variables occurring in* Γ *with* x *bound by* И *and* y *by a* Я*, or a pair of the form* $\{x^{\text{И}}, y\}$ *(resp.* $\{x^{\text{Я}}, y\}$*) made of a nominal variable* $x^{\text{И}}$ *(resp.* $x^{\text{Я}}$*) in the store, and a variable* y *occurring in* Γ *bound by the nominal quantifier* Я *(resp.* И*);*

A link is ***axiomatic*** *if it is nominal link containing no nominal variables, or a sub-sequent made of a single occurrence of* $\circ$ *or a pair of the atoms of the form* $\{\langle x!y\rangle, (z?t)\}$*. A* ***linking*** *(resp.* ***axiomatic linking****) on* $\mathcal{S} \vdash \Gamma$ *is a set of links (resp. axiomatic links) on* $\mathcal{S} \vdash \Gamma$*.*

▶ **Notation 19.** *We represent a link* a *by drawing a horizontal line labeled by* a *connected via vertical segments to the roots of each variable or subformula in the link.*

▶ **Example 20.** In the left of Equation (11) we show a decorated with the links provided on the right.

$$w^{\text{Я}}, z^{\text{И}} \vdash \text{Я}v.A \otimes B, (\text{И}x.C) \parr (D \oplus E), \text{Я}y.F \qquad \begin{array}{l} z = \{z^{\text{И}}, v\} \\ a = A, C, D \oplus E \\ b = B, \forall x.F, (\text{И}x.C) \parr (D \oplus E) \\ c = \{x, y\} \end{array} \tag{11}$$

The sequent $\Gamma' = D, D \oplus E$ is not a link for the given judgement because it is not a sub-judgement since the formula D is repeated twice.

We then fix a notation for substitutions which we use to encode the information about the witnesses of the quantifiers.

▶ **Notation 21.** *We use the standard notation* $\sigma = [x_1/y_1, \ldots, x_n/y_n]$ *for* ***substitutions****, i.e., (partial) maps over the set of variables*[7] *with* ***domain*** $\{y_1, \ldots, y_n\}$*. Moreover, we use the following denotations:*

- $\delta_{\emptyset}$ *is the* ***empty substitution****;*
- $\sigma\tau$ *is the* ***composition*** *of* σ *and* τ *(in which* σ *is applied after* τ*);*

[7] In the language of PiL we have no function symbols, but this definition could be generalized by defining a substitution as a map from variables to terms, as done in [42].

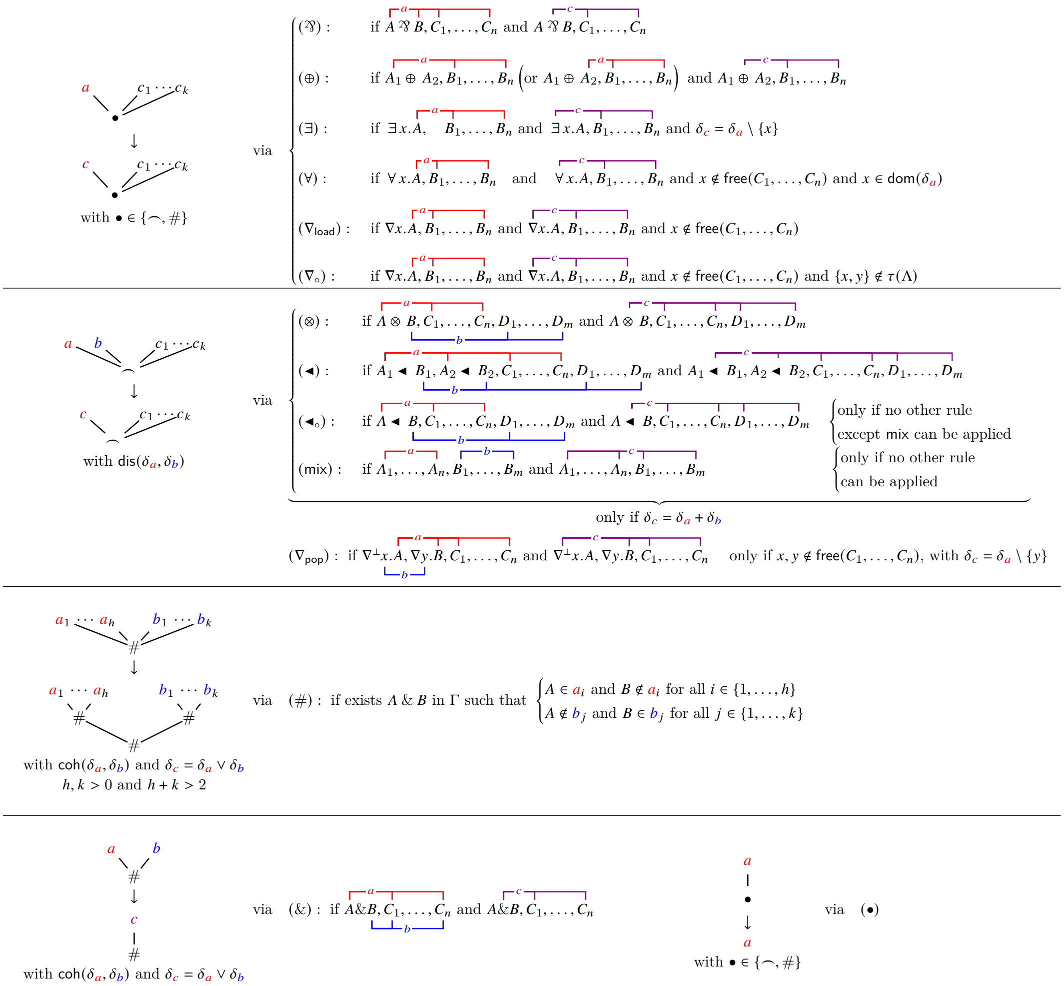


Figure 10 Coalescence steps for co-trees, with $\nabla \in \{\text{И}, \text{Я}\}$ and $a, b, c_1, \ldots, c_n$ leaves.

- $\sigma \setminus \{x\}$ *is the substitution obtained from* τ *by removing the substitution of the variable* x*;*
- σ *is* ***more general*** *than* τ *(denoted* $\sigma \leq \rho$*) if there is a map* ρ *such that* $\sigma\rho = \tau$*;*
- σ *and* τ *are* ***disjoint*** *(denoted* $\mathsf{dis}(\sigma, \tau)$*) if so are their domains. We may write* $\sigma + \tau$ *to denote* $\sigma\tau = \tau\sigma$ *whenever* $\mathsf{dis}(\sigma, \tau)$*;*
- σ *and* τ *are* ***coherent*** *(denoted* $\mathsf{coh}(\sigma, \tau)$*) if there is* ρ *such that* $\sigma\rho = \tau\rho$*;*
- *the* ***join*** *of* σ *and* τ *(denoted* $\sigma \vee \tau$*) is the least map* ρ *such that* $\sigma \leq \rho$ *and* $\tau \leq \rho$*;*

► **Definition 22.** *A* ***dualizer*** δ_a *for a link* a *on a judgement* $\mathcal{S} \vdash \Gamma$ *(or a sequent* Γ*) is a substitution with domain variables either occurring in* $\mathcal{S}$*, or occurring bound by an existential quantifier (*$\exists$*) or by a nominal quantifier (*И *or* Я*) in* Γ*.*

A ***linking with witnesses*** $\langle \Lambda, \delta^\Lambda \rangle$ *is a linking* Λ *on* $\mathcal{S} \vdash \Gamma$ *provided with a* ***witness map*** δ^Λ *associating to each link in* Λ *a (possibly empty) dualizer.*

The ***initial witness map*** *of an axiomatic linking* Λ *is defined as follows for each* $a \in \Lambda$*:*

- *if* $a = \{\langle x!y \rangle, (z?t)\}$*, then* $\delta^\Lambda_a = [x/z, y/t]$*;*
- *if* $a = \{x, y\}$ *is a nominal link with* x *bound by* И *and* y *bound by* Я*, then* $\delta^\Lambda_a = [x]y$*;*
- *if* $a = \{\circ\}$*, then* $\delta_\varnothing$*.*

5.1 Conflict Nets

Conflict nets for MALL are trees alternating *concord* ($\frown$) and *conflict* ($\#$) nodes, having the elements of an axiomatic linking as leaves, and satisfying a contractibility criterion with respect to a rewriting procedure called *coalescence*.

▶ **Definition 23.** *A* ***concord-conflict tree*** *(or* ***co-tree*** *for short) for a linking Λ on a judgement $\mathcal{S} \vdash \Gamma$ is a tree $\tau(\Lambda)$ with leaves labeled by links Λ, and internal nodes labeled by $\frown$ (**concord** nodes) or by $\#$ (**conflict** nodes). It is **axiomatic** if Λ is axiomatic. We denote by $\lfloor\tau(\Lambda)\rfloor$ the co-tree obtained by merging adjacent $\frown$-nodes (resp. $\#$-nodes), and by removing nodes with a single child (by attaching its child to its parent). A co-tree $\tau(\Lambda)$ is **canonical** if $\tau(\Lambda) = \lfloor\tau(\Lambda)\rfloor$. We may denote $\tau(\Lambda)_1 \frown \tau(\Lambda)_2$ or $\frown(\tau(\Lambda)_1, \dots, \tau(\Lambda)_n)$ (resp. $\tau(\Lambda)_1 \# \tau(\Lambda)_2$ or $\#(\tau(\Lambda)_1, \dots, \tau(\Lambda)_n)$) a co-tree with root a $\frown$-node (resp. $\#$-node) having as children the roots of $\tau(\Lambda)_1, \dots, \tau(\Lambda)_n$.*

*As for linkings with witnesses, we define a **co-tree with witnesses** $\langle\tau(\Lambda), \delta\rangle$ as a co-tree $\tau(\Lambda)$ equipped with a **witness map** δ associating to each leaf a of $\tau(\Lambda)$ a (possibly empty) dualizer δ_a.*

In order to define the coalescence criterion, we need to consider co-trees for linkings with witnesses, rather than co-trees for linkings.

▶ **Definition 24.** *In Figure 10 we define **coalescence steps** over co-trees with witnesses. A co-tree $\tau(\Lambda)$ for a linking Λ on $\vdash \Gamma$ is **coalescent** if there is a **coalescence path**, that is, a sequence of coalescence steps from the co-tree with witnesses $\langle\tau(\Lambda), \delta^{\Lambda}\rangle$ (where $\delta^{\tau(\Lambda)}$ is the initial witness map for Λ) to a co-tree with witnesses consisting of a single leaf $\vdash \Gamma$ with empty dualizer.*

*A **conflict net** for a sequent Γ is a coalescent axiomatic co-tree $\tau(\Lambda)$ on $\vdash \Gamma$.*

▶ **Theorem 25.** *Let Γ be a sequent. Then $\vdash_{\mathsf{PiL}} \Gamma$ iff there is a conflict-net $\tau(\Lambda)$ on Γ.*

Proof. For each derivation $\mathcal{D}$ of Γ, we define the (axiomatic) co-tree $\{\!\{\mathcal{D}\}\!\}_{\mathsf{co}}$ by translating top-down rules $\mathcal{D}$ as shown in Figure 11. Prove that $\tau(\Lambda)_{\mathcal{D}}$ is coalescent is trivial: it suffices to consider a coalescence path where coalescence steps, which are in correspondence with rules in PiL, respect the order in which we translate the proof – Note that rules mix and $\blacktriangleleft_\circ$ may require to be postponed during such translation, and applied out-of-order because of the side conditions we have on coalescence steps.

To prove the converse, we define *deductive co-trees* as co-trees with witnesses such that the leaves which are judgements are labeled by derivations with conclusion the corresponding judgement. For the base case, we consider the initial axiomatic co-tree $\tau(\Lambda)$ with witnesses $\delta^{\tau(\Lambda)}$ whose leaves which are judgements are labeled by rules ax or $\circ$. We then define a derivation $\mathcal{D}_{\tau(\Lambda)}$ with conclusion $\vdash \Gamma$ as shown in Figure 12, reasoning by induction on the length of a given coalescence path for $\tau(\Lambda)$, which exists by definition of conflict net. ◀

▶ **Corollary 26** (Conflict nets for MALL[1])**.** *A sequent Γ is derivable in MALL[1] if it admits a coalescent slice net $\tau(\Lambda)$ whose coalescence path only containing steps $⅋$, $\oplus$, $\exists$, $\forall$, $\otimes$, $\&$, $\#$, and $\bullet$.*

As in [52], we define the ***size*** of a proof net $\tau(\Lambda)$ on Γ as the number $|\tau(\Lambda)|$ of nodes in the co-tree $\tau(\Lambda)$ plus the number $|\Gamma|$ of nodes in the forest Γ.

▶ **Proposition 27.** *The coalescence criterion is polynomial in the size of the proof net.*

$$\left\{\!\!\left\{ \circ\,\frac{}{\mathcal{S} \vdash \circ} \right\}\!\!\right\}_{\mathsf{co}} = \left\{ \overset{a}{\circ} \right\} \qquad \left\{\!\!\left\{ \mathsf{ax}\,\frac{}{\mathcal{S} \vdash \langle x!y\rangle, (x?y)} \right\}\!\!\right\}_{\mathsf{co}} = \left\{ \overset{a}{\langle x!y\rangle\,,\,(x?y)} \right\}$$

$$\left\{\!\!\left\{ \nabla_{\mathsf{pop}}\,\frac{\begin{matrix}\mathcal{D}_1 \\ \mathcal{S} \vdash \Gamma, A[y/x]\end{matrix}}{\Sigma, y^{\nabla} \vdash \Gamma, \nabla^{\perp}x.A} \right\}\!\!\right\}_{\mathsf{co}} = \left\lfloor \{\{\mathcal{D}_1\}\}_{\mathsf{co}}\,[y/x] \frown \left\{ \overset{a}{x\,,\,y} \right\} \right\rfloor \qquad \left\{\!\!\left\{ \exists\,\frac{\begin{matrix}\mathcal{D}_1 \\ \mathcal{S} \vdash \Gamma, A[y/x]\end{matrix}}{\mathcal{S} \vdash \Gamma, \exists x.A} \right\}\!\!\right\}_{\mathsf{co}} = \{\{\mathcal{D}_1\}\}_{\mathsf{co}}\,[y/x]$$

$$\left\{\!\!\left\{ \mathsf{r}^1\,\frac{\begin{matrix}\mathcal{D}_1 \\ \mathcal{S} \vdash \Gamma_1\end{matrix}}{\mathcal{S} \vdash \Gamma} \right\}\!\!\right\}_{\mathsf{co}} = \{\{\mathcal{D}_1\}\}_{\mathsf{co}} \qquad \left\{\!\!\left\{ \nabla_{\mathsf{load}}\,\frac{\begin{matrix}\mathcal{D}_1 \\ \mathcal{S}, x^{\nabla} \vdash \Gamma, A\end{matrix}}{\mathcal{S} \vdash \Gamma, \nabla x.A} \right\}\!\!\right\}_{\mathsf{co}} = \{\{\mathcal{D}_1\}\}_{\mathsf{co}}$$

$$\left\{\!\!\left\{ \mathsf{r}^2\,\frac{\begin{matrix}\mathcal{D}_1 \\ \mathcal{S}_1 \vdash \Gamma_1\end{matrix} \quad \begin{matrix}\mathcal{D}_2 \\ \mathcal{S}_2 \vdash \Gamma_2\end{matrix}}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma} \right\}\!\!\right\}_{\mathsf{co}} = \lfloor \{\{\mathcal{D}_1\}\}_{\mathsf{co}} \frown \{\{\mathcal{D}_2\}\}_{\mathsf{co}} \rfloor \qquad \left\{\!\!\left\{ \&\,\frac{\begin{matrix}\mathcal{D}_1 \\ \mathcal{S} \vdash \Gamma_1\end{matrix} \quad \begin{matrix}\mathcal{D}_2 \\ \mathcal{S} \vdash \Gamma_2\end{matrix}}{\mathcal{S} \vdash \Gamma} \right\}\!\!\right\}_{\mathsf{co}} = \lfloor \{\{\mathcal{D}_1\}\}_{\mathsf{co}} \,\#\, \{\{\mathcal{D}_2\}\}_{\mathsf{co}} \rfloor$$

$$\text{with } \mathsf{r}^1 \in \{⅋, \oplus, \forall, И^{\circ}, Я^{\circ}\} \quad \text{and} \quad \mathsf{r}^2 \in \{\otimes, \blacktriangleleft, \blacktriangleleft_{\circ}, \mathsf{mix}\} \quad \text{and} \quad \nabla \in \{И, Я\}$$

Figure 11 Translation of a derivation in PiL into a conflict net, where $\{\{\mathcal{D}\}\}_{\mathsf{co}}\,[y/x]$ is the co-tree obtained by applying the substitution $[y/x]$ to all its links in $\{\{\mathcal{D}\}\}_{\mathsf{co}}$.

Proof. The result follows from the same argument (and algorithms) used in [52] in the proof of the similar result for MALL. The new multiplicative coalescence steps (the ones involving the ◀) are as complex as the ⊗. Coalescence steps involving quantifiers require to perform operations on dualizers which are linear in the size of the dualizer (see [61]), and the size of the dualizer is linear in the size of the formula. Thus the complexity is at most $O(n^5)$ where n is the size of the proof net. ◀

5.2 Slice Nets

In this paper we define slice nets using the correctness criterion from [53] based on *erasing steps* rather than on slicing and switchings as in [56, 55]. The criterion presented here is more similar to the criterion we used for conflict nets, it requires fewer definitions to be stated, and it is polynomial on the size of of a slice net.

▶ **Definition 28.** *The* ***parse graph***[8] *of a judgement* $\mathcal{S} \vdash \Gamma$ *is defined as the graph* $\mathcal{G}(\mathcal{S} \vdash \Gamma)$ *with:*

- *set of vertices* $\mathsf{V}(\mathcal{S} \vdash \Gamma)$ *contains the store* $\mathcal{S}$*, and the set of occurrences of atoms, connectives, quantifiers, and bound variables in the sequent* Γ*; and*
- *set of edges* $\mathsf{E}(\mathcal{S} \vdash \Gamma)$ *containing an edge* $\{v, w\}$ *whenever* v *is the main operator of a formula* $A = A_1 \odot A_2$ *(resp.* $A = \mho x.A_1$*) and* w *is the main operator of the* A_1 *or* A_2 *(resp. of the subformula* A_1 *or the variable* x*).*

The parse graph of a sequent Γ *or a formula* A *is defined similarly.*

If $\mathbb{A}$ *is a set of axiomatic linkings on* $\mathcal{S} \vdash \Gamma$*, then the* ***linked parse graph*** *is the graph* $\mathcal{G}(\mathcal{S} \vdash \Gamma :: \mathbb{A})$ *obtained by adding to* $\mathcal{G}(\mathcal{S} \vdash \Gamma)$ *the set of edges* $\mathsf{E}(\mathbb{A})$ *containing an edge between each pair of vertices corresponding to a pair of atoms or variables in link in one of the linking in* $\mathbb{A}$ *– therefore, no edge has to be considered for a link of the form* $\{\circ\}$*.*

▶ **Notation 29.** *Let* $\mathbb{A} = \{\Lambda_i\}_{i \in I}$ *be a set of atomic linkings for a judgement* $\mathcal{S} \vdash \Gamma$*. If* $\mathcal{S}' \vdash \Gamma'$ *is a sub-judgement of* $\mathcal{S} \vdash \Gamma$*, then we define* $\mathbb{A}|_{\mathcal{S}' \vdash \Gamma'}$ *as the set of non-empty linkings* $\{\Lambda'_i\}_{j \in J}$

[8] Indeed, parse graphs are always forests.

$\mathcal{D}_a$	step	$\mathcal{D}_c$
$\overset{\pi \Vert}{\delta_a(\mathcal{S} \vdash A, B, \Gamma)}$	$\overset{\parr}{\to}$	$\parr\ \dfrac{\overset{\pi \Vert}{\delta_a(\mathcal{S} \vdash A, B, \Gamma)}}{\delta_c(\mathcal{S} \vdash A \parr B, \Gamma)}$
$\overset{\pi \Vert}{\delta_a(\mathcal{S} \vdash A_i, \Gamma)}$	$\overset{\oplus}{\to}$	$\parr\ \dfrac{\overset{\pi \Vert}{\delta_a(\mathcal{S} \vdash A_i, \Gamma)}}{\delta_c(\mathcal{S} \vdash A_1 \oplus A_2, \Gamma)}$
$\overset{\pi \Vert}{\delta_a(\mathcal{S} \vdash A, \Gamma)}$	$\overset{\forall}{\to}$	$\forall\ \dfrac{\overset{\pi \Vert}{\delta_a(\mathcal{S} \vdash A, \Gamma)}}{\delta_c(\mathcal{S} \vdash \forall x.A, \Gamma)}$
$\overset{\pi \Vert}{\delta_a(\mathcal{S} \vdash A, \Gamma)}$	$\overset{\nabla_\circ}{\to}$	$\nabla_\circ\ \dfrac{\overset{\pi \Vert}{\delta_a(\mathcal{S} \vdash A, \Gamma)}}{\delta_c(\mathcal{S} \vdash \nabla x.A, \Gamma)}$
$\overset{\pi \Vert}{\delta_a(\mathcal{S} \vdash A[y/x], \Gamma)}$	$\overset{\exists}{\to}$	$\exists\ \dfrac{\overset{\pi \Vert}{\delta_a(\mathcal{S} \vdash A[y/x], \Gamma)}}{\delta_c(\mathcal{S} \vdash \exists x.A, \Gamma)}$
$\overset{\pi \Vert}{\delta_a(\mathcal{S} \vdash A[y/x], \Gamma)}$	$\overset{\nabla_{\mathsf{pop}}}{\to}$	$\nabla_{\mathsf{pop}}\ \dfrac{\overset{\pi \Vert}{\delta_a(\mathcal{S} \vdash A[y/x], \Gamma)}}{\delta_c(\mathcal{S}, x^\nabla \vdash \nabla^\perp x.A, \Gamma)}$
$\overset{\pi \Vert}{\delta_a(\mathcal{S}, x^\nabla \vdash A, B, \Gamma)}$	$\overset{\nabla_{\mathsf{load}}}{\to}$	$\nabla_{\mathsf{load}}\ \dfrac{\overset{\pi \Vert}{\delta_a(\mathcal{S}, x^\nabla \vdash A, \Gamma)}}{\delta_c(\mathcal{S} \vdash \nabla x.A, \Gamma)}$

with $\delta_c = \begin{cases} \delta_a \setminus \{x\} & \text{for steps } \exists \text{ and } \nabla_{\mathsf{pop}}, \\ \delta_a & \text{otherwise} \end{cases}$

$\mathcal{D}_a$	$\mathcal{D}_b$	step	$\mathcal{D}_c$
$\overset{\pi_1 \Vert}{\delta_a(\mathcal{S}_1 \vdash A, \Gamma)}$	$\overset{\pi_2 \Vert}{\delta_b(\mathcal{S}_2 \vdash B, \Delta)}$	$\overset{\otimes}{\to}$	$\otimes\ \dfrac{\overset{\pi_1 \Vert}{\delta_a(\mathcal{S}_1 \vdash A, \Gamma)} \quad \overset{\pi_2 \Vert}{\delta_b(\mathcal{S}_2 \vdash B, \Delta)}}{\delta_c(\mathcal{S}_1, \mathcal{S}_2 \vdash A \otimes B, \Gamma, \Delta)}$
$\overset{\pi_1 \Vert}{\delta_a(\mathcal{S}_1 \vdash A, \Gamma)}$	$\overset{\pi_2 \Vert}{\delta_b(\mathcal{S}_2 \vdash B, \Delta)}$	$\overset{\blacktriangleleft_\circ}{\to}$	$\blacktriangleleft_\circ\ \dfrac{\overset{\pi_1 \Vert}{\delta_a(\mathcal{S}_1 \vdash A, \Gamma)} \quad \overset{\pi_2 \Vert}{\delta_b(\mathcal{S}_2 \vdash B, \Delta)}}{\delta_c(\mathcal{S}_1, \mathcal{S}_2 \vdash A \blacktriangleleft B, \Gamma, \Delta)}$
$\overset{\pi_1 \Vert}{\delta_a(\mathcal{S}_1 \vdash A_1, A_2\Gamma)}$	$\overset{\pi_2 \Vert}{\delta_b(\mathcal{S}_2 \vdash B_1, B_2\Delta)}$	$\overset{\blacktriangleleft}{\to}$	$\blacktriangleleft\ \dfrac{\overset{\pi_1 \Vert}{\delta_a(\mathcal{S}_1 \vdash A_1, A_2, \Gamma)} \quad \overset{\pi_2 \Vert}{\delta_b(\mathcal{S}_2 \vdash B_1, B_2, \Delta)}}{\delta_c(\mathcal{S}_1, \mathcal{S}_2 \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, \Gamma, \Delta)}$
$\overset{\pi_1 \Vert}{\delta_a(\mathcal{S}_1 \vdash \Gamma)}$	$\overset{\pi_2 \Vert}{\delta_b(\mathcal{S}_2 \vdash \Delta)}$	$\overset{\mathsf{mix}}{\to}$	$\mathsf{mix}\ \dfrac{\overset{\pi_1 \Vert}{\delta_a(\mathcal{S}_1 \vdash \Gamma)} \quad \overset{\pi_2 \Vert}{\delta_b(\mathcal{S}_2 \vdash \Delta)}}{\delta_c(\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta)}$
$\overset{\pi_1 \Vert}{\delta_a(\mathcal{S} \vdash A, \Gamma)}$	$\overset{\pi_2 \Vert}{\delta_b(\mathcal{S} \vdash B, \Delta)}$	$\overset{\&}{\to}$	$\&\ \dfrac{\overset{\pi_1 \Vert}{\delta_a(\mathcal{S} \vdash A, \Gamma)} \quad \overset{\pi_2 \Vert}{\delta_b(\mathcal{S} \vdash B, \Delta)}}{\delta_c(\mathcal{S} \vdash A \,\&\, B, \Gamma, \Delta)}$

with $\delta_c = \begin{cases} \delta_a \vee \delta_b & \text{for step } \& \\ \delta_a + \delta_b & \text{otherwise} \end{cases}$

Figure 12 Effect of coalescence steps in Figure 10 on co-trees with leaves labeled by derivations. The steps • and # change no link labels.

(for a $J \subseteq I$) such that each linking Λ'_i contains all links in Λ_i which are pairs of vertices in $\mathsf{V}(\mathcal{S}' \vdash \Gamma')$*, that is,* $\mathbb{A}|_{\mathcal{S}' \vdash \Gamma'} = \left\{ \Lambda_i|_{\mathcal{S}' \vdash \Gamma'} := \{\lambda \in \Lambda_i \mid \lambda \subseteq \mathsf{V}(\mathcal{S}' \vdash \Gamma')\} \mid i \in I,\ \Lambda_i|_{\mathcal{S}' \vdash \Gamma'} \neq \varnothing \right\}$.

► **Definition 30.** *In Figure 13 we define **erasing steps** as the rewriting rules over sets of linkings with witnesses*[9]*. We say that a set of axiomatic linkings $\mathbb{A}$ on a sequent Γ is **erasable** if there is a sequence of erasing steps (called **erasing path**) starting from the singleton* $\{\mathcal{G}\,(\varnothing \vdash \Gamma :: \mathbb{A}, \delta^{\mathbb{A}})\}$ *and ending with the singleton containing* $\varnothing = \mathcal{G}\,(\varnothing \vdash \varnothing :: \varnothing, \delta_\varnothing)$*.*

*A **slice net** for Γ is an erasable set of axiomatic linkings $\mathbb{A}$.*

► Remark 31. The side conditions of our erasing steps ensures that the connective(s) removed by the step is “ready” to be sequentialized, similarly to how the readiness is defined in [53] for the slice nets for MALL.

In Figure 14 we show an erasing path for the slice net from Figure 1 in the introduction.

► **Theorem 32.** *Let Γ be a non-empty sequent. Then $\vdash_{\mathsf{PiL}} \Gamma$ iff there is a slice net $\mathbb{A}$ on Γ.*

Proof. In Figure 15 we provide translation from derivations to slice nets defined inductively on the structure of the derivation. The obtained set of axiomatic linkings is erasable by definition, since each inductive step of the translation preserve erasability.

As for conflict nets, sequentialization for slice nets follows from the existence of a derivation constructed from a given erasing path, since each erasing step corresponds to an application of a sequent rule (with the same name). ◄

► **Corollary 33** (Slice nets for MALL[1]). *A sequent Γ is derivable in* MALL[1] *if it admits an erasable slice net $\tau(\Lambda)$ whose erasing path only containing steps* ax, $\parr$, $\oplus$, $\exists$, $\forall$, $\otimes$, *and* $\&$*.*

► **Corollary 34.** *The erasing criterion is polynomial in the size of the slice net.*

[9] We represent erasing steps without explicitly writing the context, that is, if $L \to_{\mathsf{er}} R_1, \ldots, R_k$ is a rule, then it can be applied in a set in such a way $\{L, x_1, \ldots, x_n\}$ can be rewritten as $\{R_1 \ldots, R_k, x_1, \ldots, x_n\}$.

Erasing rule	Side conditions
$\mathcal{G}\left(\mathcal{S} \vdash \langle x!y\rangle, (z?t) :: \{a\}\right)$ $\downarrow_{\mathsf{er}}$ via ax	$a = \{\langle x!y\rangle, (z?t)\}$ with $\delta_a^{\{a\}} = [y/x, t/z]$
$\mathcal{G}\left(\mathcal{S} \vdash \circ :: \{\varnothing\}\right)$ $\downarrow_{\mathsf{er}}$ via $\circ$	
$\mathcal{G}\left(\mathcal{S} \vdash \Gamma, A \parr B :: \mathbb{A}\right)$ $\downarrow_{\mathsf{er}}$ via $\parr$ $\mathcal{G}\left(\mathcal{S} \vdash \Gamma, A, B :: \mathbb{A}\right)$	
$\mathcal{G}\left(\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta :: \mathbb{A}\right)$ $\downarrow_{\mathsf{er}}$ via mix $\mathcal{G}\left(\mathcal{S}_1 \vdash \Gamma :: \mathbb{A}\vert_{\mathcal{S}_1 \vdash \Gamma}\right), \mathcal{G}\left(\mathcal{S}_2 \vdash \Delta :: \mathbb{A}\vert_{\mathcal{S}_2 \vdash \Delta}\right)$	if there is no $\{v, w\} \in \mathsf{E}(\mathbb{A})$ such that $v \in \mathsf{V}(\mathcal{S}_1 \vdash \Gamma)$ and $w \in \mathsf{V}(\mathcal{S}_2 \vdash \Delta)$
$\mathcal{G}\left(\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta, A \blacktriangleleft B, C \blacktriangleleft D :: \mathbb{A}\right)$ $\downarrow_{\mathsf{er}}$ via $\blacktriangleleft$ $\mathcal{G}\left(\mathcal{S}_1 \vdash \Gamma, A, C :: \mathbb{A}\vert_{\mathcal{S}_1 \vdash \Gamma, A}\right), \mathcal{G}\left(\mathcal{S}_2 \vdash \Delta, B, D :: \mathbb{A}\vert_{\mathcal{S}_2 \vdash \Delta, B}\right)$	$\mathcal{G}\left(\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta, A \blacktriangleleft B, C \blacktriangleleft D :: \mathbb{A}\right)$ is connected and $\mathcal{G}\left(\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta, A, B, C, D :: \mathbb{A}\right)$ is not connected
$\mathcal{G}\left(\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta, A \blacktriangleleft B :: \mathbb{A}\right)$ $\downarrow_{\mathsf{er}}$ via $\blacktriangleleft_\circ$ $\mathcal{G}\left(\mathcal{S}_1 \vdash \Gamma, A :: \mathbb{A}\vert_{\mathcal{S}_1 \vdash \Gamma, A}\right), \mathcal{G}\left(\mathcal{S}_2 \vdash \Delta, B :: \mathbb{A}\vert_{\mathcal{S}_2 \vdash \Delta, B}\right)$	$\mathcal{G}\left(\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta, A \blacktriangleleft B :: \mathbb{A}\right)$ is connected and $\mathcal{G}\left(\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta, A, B :: \mathbb{A}\right)$ is not connected
$\mathcal{G}\left(\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta, A \otimes B :: \mathbb{A}\right)$ $\downarrow_{\mathsf{er}}$ via $\otimes$ $\mathcal{G}\left(\mathcal{S}_1 \vdash \Gamma, A :: \mathbb{A}\vert_{\mathcal{S}_1 \vdash \Gamma, A}\right), \mathcal{G}\left(\mathcal{S}_2 \vdash \Delta, B :: \mathbb{A}\vert_{\mathcal{S}_2 \vdash \Delta, B}\right)$	$\mathcal{G}\left(\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta, A \otimes B :: \mathbb{A}\right)$ is connected and $\mathcal{G}\left(\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta, A, B :: \mathbb{A}\right)$ is not connected
$\mathcal{G}\left(\mathcal{S} \vdash \Gamma, A_1 \oplus A_2 :: \mathbb{A}\right)$ $\downarrow_{\mathsf{er}}$ via $\oplus$ $\mathcal{G}\left(\mathcal{S} \vdash \Gamma, A_i :: \mathbb{A}\right)$	if no vertex in $\mathsf{V}(A_{1-i})$ occurs in $\mathsf{E}(\mathbb{A})$ for $i \in \{1,2\}$
$\mathcal{G}\left(\mathcal{S} \vdash \Gamma, A_1 \,\&\, A_2 :: \mathbb{A}_1 \uplus \mathbb{A}_2\right)$ $\downarrow_{\mathsf{er}}$ via $\&$ $\mathcal{G}\left(\mathcal{S} \vdash \Gamma, A_1 :: \mathbb{A}_1\right), \mathcal{G}\left(\mathcal{S} \vdash \Gamma, A_2 :: \mathbb{A}_2\right)$	if no vertex in $\mathsf{V}(A_{1-i})$ occurs in $\mathsf{E}(\mathbb{A}_i)$ for $i \in \{1,2\}$
$\mathcal{G}\left(\mathcal{S} \vdash \Gamma, \exists x.A :: \mathbb{A}\right)$ $\downarrow_{\mathsf{er}}$ via $\exists$ $\mathcal{G}\left(\mathcal{S} \vdash \Gamma, A[y/x] :: \mathbb{A}\right)$	if x occurs in a $v \in \mathsf{V}(A)$ and $v \in a \in \Lambda \in \mathbb{A}$, then $\delta_a^{\mathbb{A}}(x) = \delta_a^{\mathbb{A}}(y)$
$\mathcal{G}\left(\mathcal{S} \vdash \Gamma, \forall x.A :: \mathbb{A}\right)$ $\downarrow_{\mathsf{er}}$ via $\forall$ $\mathcal{G}\left(\mathcal{S} \vdash \Gamma, A :: \mathbb{A}\right)$	if $x \notin \mathsf{free}(\Gamma)$
$\mathcal{G}\left(\mathcal{S} \vdash \Gamma, \nabla x.A :: \mathbb{A}\right)$ $\downarrow_{\mathsf{er}}$ via $\nabla_\circ$ $\mathcal{G}\left(\mathcal{S} \vdash \Gamma, A :: \mathbb{A}\right)$	if $x \notin \mathsf{free}(\Gamma)$ and x does not occur in a nominal link in any $\Lambda \in \mathbb{A}$
$\mathcal{G}\left(\mathcal{S} \vdash \Gamma, \nabla x.A :: \mathbb{A}\right)$ $\downarrow_{\mathsf{er}}$ via ∇_{load} $\mathcal{G}\left(\mathcal{S}, x^{\nabla} \vdash \Gamma, A :: \mathbb{A}[\{x^{\nabla}, y\}/\{x, y\}]\right)$	if $x \notin \mathsf{free}(\Gamma)$ and $\{x, y\} \in \mathsf{E}(\mathbb{A})$
$\mathcal{G}\left(\mathcal{S}, y^{\nabla} \vdash \Gamma, \nabla^{\perp} x.A :: \mathbb{A}\right)$ $\downarrow_{\mathsf{er}}$ via ∇_{pop} $\mathcal{G}\left(\mathcal{S} \vdash \Gamma, A[y/a] :: \mathbb{A} \setminus a\right)$	$a = \{x, y^{\nabla}\} \in \mathsf{E}(\mathbb{A})$ where $\mathbb{A} \setminus a := \{\Lambda \setminus \{a\} \mid \Lambda \in \mathbb{A}\}$

Figure 13 Erasing steps for linked parse graphs.

Proof. The length of a successful erasing path is bound by the size of Γ, and checking and applying each erasing step can be performed in polinomial time $\mathcal{G}(\mathbb{A})$ ◀

▶ Remark 35. By case analysis, it is easy to see that erasing steps are locally confluent.

▶ Remark 36. The proof translation from derivations to slice nets could be equivalently defined by letting $\mathbb{A}(\mathcal{D})$ to be the set of axiomatic linkings on Γ defined from the conflict net $\{\!\{\mathcal{D}\}\!\}_{\mathsf{co}}$ as follows:

$$\{\!\{\tau(\Lambda)\}\!\}_{\mathsf{sl}} = \begin{cases} \{a\} & \text{if } a \text{ is an axiom link} \\ X_1 \cup \cdots \cup X_n & \text{if } \mathcal{D} = \#(X_1, \ldots, X_n) \\ \left\{ \bigcup_{x_1 \in \{\!\{X_1\}\!\}_{\mathsf{sl}}} \cdots \bigcup_{x_n \in \{\!\{X_n\}\!\}_{\mathsf{sl}}} (x_1 \cup \cdots \cup x_n) \right\} & \text{if } \mathcal{D} = \frown(X_1, \ldots, X_n) \end{cases}$$

In this translation, it is clear that each conflict node in $\{\!\{\mathcal{D}\}\!\}_{\mathsf{co}}$ multiplies the elements in the set of linkings, leading to a non-polinomial time transation.

$$\left\{\mathcal{G}\left(\vdash \text{И}x.\text{И}y.\left(\langle y!a\rangle \parr \exists a.(y?a) \parr \begin{pmatrix}\langle x!\ell_1\rangle \blacktriangleleft \exists b.(x?b)\\ \&\\ \langle x!\ell_2\rangle \blacktriangleleft \langle x!c\rangle\end{pmatrix} \parr \begin{pmatrix}(x?\ell_1) \blacktriangleleft \langle x!b\rangle\\ \oplus\\ (x?\ell_2) \blacktriangleleft \exists c.(x?c)\end{pmatrix}\right) :: \mathbb{A}\right)\right\}$$

$$\downarrow_{\mathsf{er}} \text{ via } (2\times \text{И}_\circ) + (2\times \parr)$$

$$\left\{\mathcal{G}\left(\vdash \langle y!a\rangle, (y?a), \begin{pmatrix}\langle x!\ell_1\rangle \blacktriangleleft \exists b.(x?b)\\ \&\\ \langle x!\ell_2\rangle \blacktriangleleft \langle x!c\rangle\end{pmatrix}, \begin{pmatrix}(x?\ell_1) \blacktriangleleft \langle x!b\rangle\\ \oplus\\ (x?\ell_2) \blacktriangleleft \exists c.(x?c)\end{pmatrix} :: \mathbb{A}\right)\right\}$$

$$\downarrow_{\mathsf{er}} \text{ via } \mathsf{mix}$$

$$\left\{\begin{array}{c}\mathcal{G}\left(\vdash \langle y!a\rangle, (y?a) :: \{\{\langle y!a\rangle, (y?a)\}\}\right),\\ \mathcal{G}\left(\vdash \begin{pmatrix}\langle x!\ell_1\rangle \blacktriangleleft \exists b.(x?b)\\ \&\\ \langle x!\ell_2\rangle \blacktriangleleft \langle x!c\rangle\end{pmatrix}, \begin{pmatrix}(x?\ell_1) \blacktriangleleft \langle x!b\rangle\\ \oplus\\ (x?\ell_2) \blacktriangleleft \exists c.(x?c)\end{pmatrix} :: \left\{\begin{array}{c}\left\{\begin{array}{c}\{\langle x!b\rangle, (x?b)\},\\ \{\langle x!\ell_1\rangle, (x?\ell_1)\}\end{array}\right\},\\ \left\{\begin{array}{c}\{\langle x!c\rangle, (x?c)\},\\ \{\langle x!\ell_2\rangle, (x?\ell_2)\}\end{array}\right\}\end{array}\right\}\right)\end{array}\right\}$$

$$\downarrow_{\mathsf{er}} \text{ via } \mathsf{ax}$$

$$\left\{\mathcal{G}\left(\vdash \begin{pmatrix}\langle x!\ell_1\rangle \blacktriangleleft \exists b.(x?b)\\ \&\\ \langle x!\ell_2\rangle \blacktriangleleft \langle x!c\rangle\end{pmatrix}, \begin{pmatrix}(x?\ell_1) \blacktriangleleft \langle x!b\rangle\\ \oplus\\ (x?\ell_2) \blacktriangleleft \exists c.(x?c)\end{pmatrix} :: \left\{\begin{array}{c}\left\{\begin{array}{c}\{\langle x!b\rangle, (x?b)\},\\ \{\langle x!\ell_1\rangle, (x?\ell_1)\}\end{array}\right\},\\ \left\{\begin{array}{c}\{\langle x!c\rangle, (x?c)\},\\ \{\langle x!\ell_2\rangle, (x?\ell_2)\}\end{array}\right\}\end{array}\right\}\right)\right\}$$

$$\downarrow_{\mathsf{er}} \text{ via } \& + (2\times\oplus)$$

$$\left\{\begin{array}{c}\mathcal{G}\left(\vdash \langle x!\ell_1\rangle \blacktriangleleft \exists b.(x?b), (x?\ell_1) \blacktriangleleft \langle x!b\rangle :: \left\{\left\{\begin{array}{c}\{\langle x!b\rangle, (x?b)\},\\ \{\langle x!\ell_1\rangle, (x?\ell_1)\}\end{array}\right\}\right\}\right),\\ \mathcal{G}\left(\vdash \langle x!\ell_2\rangle \blacktriangleleft \langle x!c\rangle, (x?\ell_2) \blacktriangleleft \exists c.(x?c) :: \left\{\left\{\begin{array}{c}\{\langle x!c\rangle, (x?c)\},\\ \{\langle x!\ell_2\rangle, (x?\ell_2)\}\end{array}\right\}\right\}\right)\end{array}\right\}$$

$$\downarrow_{\mathsf{er}} \text{ via } 2\times \blacktriangleleft$$

$$\left\{\begin{array}{c}\mathcal{G}\left(\vdash \langle x!\ell_1\rangle, (x?\ell_1) :: \{\{\{\langle x!b\rangle, (x?b)\}\}\}\right),\\ \mathcal{G}\left(\vdash \exists b.(x?b), \langle x!b\rangle :: \{\{\{\langle x!\ell_1\rangle, (x?\ell_1)\}\}\}\right),\\ \mathcal{G}\left(\vdash \langle x!\ell_2\rangle, (x?\ell_2) :: \{\{\{\langle x!\ell_2\rangle, (x?\ell_2)\}\}\}\right),\\ \mathcal{G}\left(\vdash \langle x!c\rangle, \exists c.(x?c) :: \{\{\{\langle x!c\rangle, (x?c)\},\}\}\right)\end{array}\right\}$$

$$\downarrow_{\mathsf{er}} \text{ via } 4\times\mathsf{ax}$$

$$\{\{\mathcal{G}\,(\vdash \varnothing :: \varnothing)\}\}$$

Figure 14 A possible erasing path for the slice net from Figure 1.

6 Canonicity results

In this section we prove that conflict nets and slice nets for PiL are proof systems with two distinct canonicity properties called *local* and *strong* canonicity, and they both identify derivations modulo what we refer to as *witness renaming*.

We first introduce three notions of equivalence for derivations in PiL.

▶ **Definition 37.** *We call the variable introduced during the proof search by a quantifier rule the* ***active variable*** *of that (occurrence of) rule. The active variable of an existential (resp. universal) quantifier may also be called its* ***witness*** [10] *(resp.* ***eigenvariable****), while the active formula of a nominal quantifier may also be called a* ***fresh name****. Two derivations $\mathcal{D}_1$ and $\mathcal{D}_2$ in* PiL *are:*

- ***equivalent modulo active variables*** *(denoted $\mathcal{D}_1 \sim_{\mathsf{w}} \mathcal{D}_2$) if it is possible to transform $\mathcal{D}_1$ into $\mathcal{D}_2$ by changing the active variables of the quantifier rules (and propagating the changes upwards in the derivation);*

[10] In the general setting of first-order logic, the witness of an existential quantifier could be any term of the language. However, in PiL witnesses can only be variables because of the simple structure of terms.

$$\left\{\!\!\left\{ \frac{}{\mathcal{S} \vdash \circ}\circ \right\}\!\!\right\}_{\mathsf{sl}} = \left\{\!\!\left\{ \overset{a}{\circ} \right\}\!\!\right\} \qquad \left\{\!\!\left\{ \mathsf{ax}\frac{}{\mathcal{S} \vdash \langle x!y\rangle, (x?y)} \right\}\!\!\right\}_{\mathsf{sl}} = \left\{\!\!\left\{ \overset{a}{\langle x!y\rangle \, , \, (x?y)} \right\}\!\!\right\}$$

$$\left\{\!\!\left\{ \nabla_{\mathsf{pop}}\frac{\overset{\mathcal{D}_1}{\mathcal{S} \vdash \Gamma, A[y/x]}}{\Sigma, y^{\nabla} \vdash \Gamma, \nabla^{\perp}x.A} \right\}\!\!\right\}_{\mathsf{sl}} = \left\{ \Lambda \cup \left\{\!\!\left\{ \overset{a}{x \, , \, y} \right\}\!\!\right\} \mid \Lambda \in \{\!\{\mathcal{D}_1\}\!\}_{\mathsf{sl}}\,[y/x] \right\} \qquad \left\{\!\!\left\{ \exists\frac{\overset{\mathcal{D}_1}{\mathcal{S} \vdash \Gamma, A[y/x]}}{\mathcal{S} \vdash \Gamma, \exists x.A} \right\}\!\!\right\}_{\mathsf{sl}} = \{\!\{\mathcal{D}_1\}\!\}_{\mathsf{sl}}\,[y/x]$$

$$\left\{\!\!\left\{ \mathsf{r}^1\frac{\overset{\mathcal{D}_1}{\mathcal{S} \vdash \Gamma_1}}{\mathcal{S} \vdash \Gamma} \right\}\!\!\right\}_{\mathsf{sl}} = \{\!\{\mathcal{D}_1\}\!\}_{\mathsf{sl}} \qquad \left\{\!\!\left\{ \nabla_{\mathsf{load}}\frac{\overset{\mathcal{D}_1}{\mathcal{S}, x^{\nabla} \vdash \Gamma, A}}{\mathcal{S} \vdash \Gamma, \nabla x.A} \right\}\!\!\right\}_{\mathsf{sl}} = \{\!\{\mathcal{D}_1\}\!\}_{\mathsf{sl}}$$

$$\left\{\!\!\left\{ \mathsf{r}^2\frac{\overset{\mathcal{D}_1}{\mathcal{S}_1 \vdash \Gamma_1} \quad \overset{\mathcal{D}_2}{\mathcal{S}_2 \vdash \Gamma_2}}{\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma} \right\}\!\!\right\}_{\mathsf{sl}} = \{\Lambda_1 \cup \Lambda_2 \mid \Lambda_1 \in \{\!\{\mathcal{D}_1\}\!\}_{\mathsf{sl}}, \Lambda_2 \in \{\!\{\mathcal{D}_2\}\!\}_{\mathsf{sl}}\} \qquad \left\{\!\!\left\{ \&\frac{\overset{\mathcal{D}_1}{\mathcal{S} \vdash \Gamma_1} \quad \overset{\mathcal{D}_2}{\mathcal{S} \vdash \Gamma_2}}{\mathcal{S} \vdash \Gamma} \right\}\!\!\right\}_{\mathsf{sl}} = \{\!\{\mathcal{D}_1\}\!\}_{\mathsf{sl}} \cup \{\!\{\mathcal{D}_2\}\!\}_{\mathsf{sl}}$$

with $\mathsf{r}^1 \in \{\invamp, \oplus, \forall, И^{\circ}, Я^{\circ}\}$ and $\mathsf{r}^2 \in \{\otimes, \blacktriangleleft, \blacktriangleleft_{\circ}, \mathsf{mix}\}$ and $\nabla \in \{И, Я\}$

Figure 15 Translation of a derivation in PiL into a slice net, where $\{\!\{\mathcal{D}\}\!\}_{\mathsf{co}}\,[y/x]$ is the set of linkings obtained by applying the substitution $[y/x]$ to all its links in them, and by letting the dualizers δ_a in $\{\!\{\mathcal{D}\}\!\}_{\mathsf{sl}}$ being $\delta_a[y/x]$ in $\{\!\{\mathcal{D}\}\!\}_{\mathsf{sl}}\,[y/x]$.

- ***equivalent modulo rule permutations*** *(denoted $\mathcal{D}_1 \simeq \mathcal{D}_2$) if it is possible to transform $\mathcal{D}_1$ into $\mathcal{D}_2$ using all transformations in Figure 16;*
- ***equivalent modulo local rule permutations*** *(denoted $\mathcal{D}_1 \approx \mathcal{D}_2$) if it is possible to transform $\mathcal{D}_1$ into $\mathcal{D}_2$ using the local rule permutations in Figure 16.*

Moreover, we write $\mathcal{D}_1 \simeq_{\mathsf{w}} \mathcal{D}_2$ (resp. $\mathcal{D}_1 \approx_{\mathsf{w}} \mathcal{D}_2$) if they are equivalent modulo rule permutations (resp. local rule permutations) and active variables, that is, if there are derivations $\mathcal{D}_1'$ and $\mathcal{D}_2''$ such that $\mathcal{D}_1 \sim_{\mathsf{w}} \mathcal{D}_1' \simeq \mathcal{D}_2' \sim_{\mathsf{w}} \mathcal{D}_2$ (resp. $\mathcal{D}_1 \sim_{\mathsf{w}} \mathcal{D}_1' \approx \mathcal{D}_2' \sim_{\mathsf{w}} \mathcal{D}_2$).

▶ Remark 38. In Equation (12), we show three derivations which are equivalent modulo active variables.

$$И_{\mathsf{load}}\frac{И_{\mathsf{pop}}\dfrac{\mathsf{ax}\dfrac{}{\vdash \langle x!a\rangle, (x?a)}}{x^{И} \vdash \langle x!a\rangle, Яy.(y?a)}}{\vdash Иx.\langle x!a\rangle, Яy.(y?a)} \sim_{\mathsf{w}} И_{\mathsf{load}}\frac{И_{\mathsf{pop}}\dfrac{\mathsf{ax}\dfrac{}{\vdash \langle z!a\rangle, (z?a)}}{z^{И} \vdash \langle z!a\rangle, Яy.(y?a)}}{\vdash Иx.\langle x!a\rangle, Яy.(y?a)} \sim_{\mathsf{w}} И_{\mathsf{load}}\frac{И_{\mathsf{pop}}\dfrac{\mathsf{ax}\dfrac{}{\vdash \langle y!a\rangle, (y?a)}}{y^{И} \vdash \langle y!a\rangle, Яy.(y?a)}}{\vdash Иx.\langle x!a\rangle, Яy.(y?a)} \tag{12}$$

In Equation (13) we show two existential quantifier rules select two distinct witnesses x and z, but the pair of atoms linked by an axiom rule is the same.

$$\exists\frac{\mathsf{ax}\dfrac{}{\vdash \langle x!a\rangle, (x?a)}}{\vdash \exists x.\langle x!a\rangle, \exists y.(y?a)} \sim_{\mathsf{w}} \exists\frac{\mathsf{ax}\dfrac{}{\vdash \langle y!a\rangle, (y?a)}}{\vdash \exists x.\langle x!a\rangle, \exists y.(y?a)} \tag{13}$$

We could argue that these two derivations should be not identified because the choice of the witness is part of the information of the proof. In a boarder sense, it may be useful to not identify a proof using a very elementary witness with a proof using a quite complex one. However, because of the quite limited syntax of atoms (terms) in PiL, witness for quanfitiers can only be varibles. Therefore, as soon as the choice of witness do not change the pairs of atoms which are mated by the ax-rules in the derivation, such a choice can be considered irrelevant, especially if the choice of the witness of a ∇_{pop} depends on varible previously stored by a ∇_{load}, which is arbitrary because of α-equivalence. Note that the two

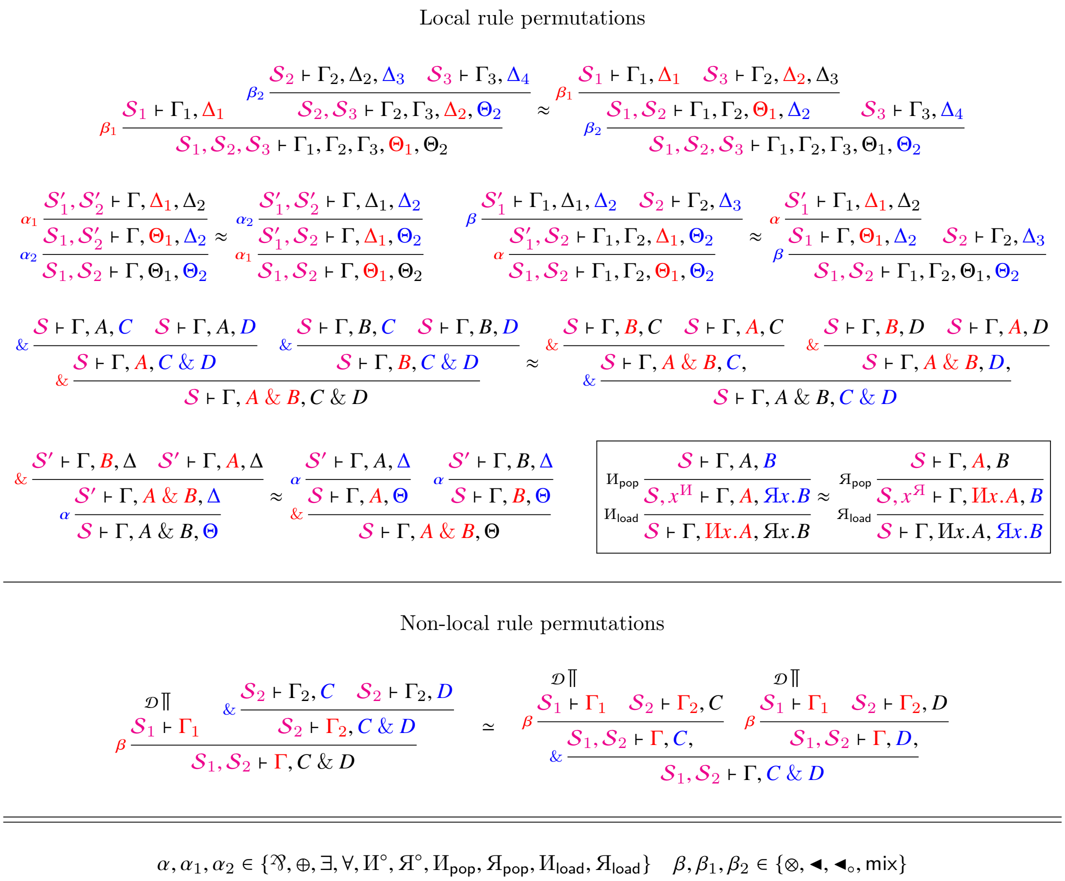
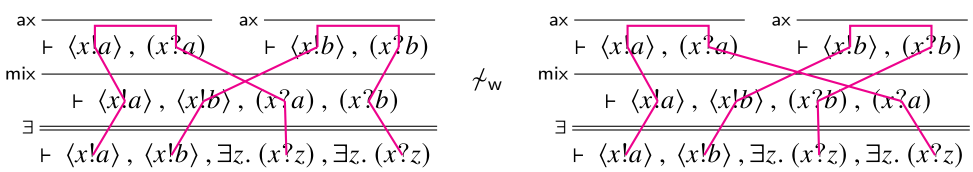


Figure 16 Rule permutations in PiL.

sub-derivations of the $\sim_{\mathsf{w}}$-equivalent derivations in Equation (13) made only of the ax-rules are not $\sim_{\mathsf{w}}$-equivalent. That is $\mathsf{ax}\dfrac{}{\vdash \langle x!a\rangle, (x?a)} \not\sim_{\mathsf{w}} \mathsf{ax}\dfrac{}{\vdash \langle y!a\rangle, (y?a)}$.

However, the choice of active variables may change the pair of atoms linked by the ax-rules. For an example, see Equation (14) below, where we show two non $\sim_{\mathsf{w}}$-equivalent derivations in which we trace the occurrences of atoms in the derivation to show how the choice of the active variables changes the pairs of atoms linked by the ax-rules.



(14)

We conclude by proving our the canonicity results.

▶ **Definition 39.** *Two conflict nets $\tau(\Lambda)_1$ and $\tau(\Lambda)_2$ are* ***the same*** *(denoted $\tau(\Lambda)_1 = \tau(\Lambda)_2$) if they are isomorphic co-tree, that is, if there is a bijection between the nodes of $\tau(\Lambda)_1$ and $\tau(\Lambda)_2$ that preserves the structure of the tree.*

For slice nets $\mathbb{A}_1$ and $\mathbb{A}_2$ are ***the same*** *(denoted $\mathbb{A}_1 = \mathbb{A}_2$) if they are the same set of linkings.*

▶ **Lemma 40.** *Let $\tau(\Lambda)$ be a conflict net on Γ such that $\tau(\Lambda)$ sequetializes to $\mathcal{D}_{\tau(\Lambda)}$. If $\tau(\Lambda) \to^* \tau(\Lambda)'$, then $\tau(\Lambda)'$ sequentializes to a $\mathcal{D}_{\tau(\Lambda)'}$ such that $\mathcal{D}_{\tau(\Lambda)'} \approx_{\mathsf{w}} \mathcal{D}_{\tau(\Lambda)}$.*

Proof. It suffices to check that each critical pair of coalescence steps on deductive co-trees converges. For this, we should consider the derivations labeling each link modulo the proof equivalence generated by local rule permutations Figure 16.

The most convoluted case is the one for the pair $\#/\#$, for which we use the same argument in [42], remarking that the base case of induction presents no problems in our setting (where we have the additional information of the dualizers) because of the associativity of the join operator on dualizers. Note that critical pairs with non-trivial confluence (i.e., non-local) are the ones where the rule & interacts with quantifiers, and the ones of the form mix/mix and $◀_\circ / ◀_\circ$. Details are provided in Appendix B. ◀

▶ **Theorem 41.** *Let $\mathcal{D}$ and $\mathcal{D}'$ be two derivations. Then, the following hold:*
1. *$\mathcal{D} \approx_{\mathsf{w}} \mathcal{D}'$ iff $\{\{\mathcal{D}\}\}_{\mathsf{co}} = \{\{\mathcal{D}'\}\}_{\mathsf{co}}$.*
2. *$\mathcal{D} \simeq_{\mathsf{w}} \mathcal{D}'$ iff $\{\{\mathcal{D}\}\}_{\mathsf{sl}} = \{\{\mathcal{D}'\}\}_{\mathsf{sl}}$.*

Proof. It suffices to consider the case analysis given by the rule permutations in Figure 16 as done in [52] for conflict nets, and in [56] for slice nets.
1. For $\approx_{\mathsf{w}}$ it follows from the definition of $\{\{\cdot\}\}_{\mathsf{co}}$ and by Lemma 40. The only new cases with respect to [52] are given by the local rule permutation changing $И_{\mathsf{pop}} + И_{\mathsf{load}}$ into a $Я_{\mathsf{pop}} + Я_{\mathsf{load}}$ (the permutation boxed in Figure 16), and the definition of equivalence modulo fresh names.
2. Similarly, for $\simeq_{\mathsf{w}}$ it follows from the definition of $\{\{\cdot\}\}_{\mathsf{sl}}$ and by Remark 35.

◀

7 Processes as Formulas

In this section we show how PiL can be used as a logical framework in which we can intepret proofs as execution trees in the π-calculus.

First we provide a translation from processes to formulas in PiL and show that the logical implication in PiL captures the structural congruence in the π-calculus. According to [64], the absence of this property suggests that the logical framework may lack a robust design.

▶ **Notation 42.** *Because of the monoidal laws, we consider generalized n-ary versions (with $n > 0$) of the additive connectives $\oplus$ and &, which are more convenient for the translation of processes. Their inference rules are defined as expected: the n-ary version of the $\oplus$-rule keeps a unique component of the n-ary disjunction, while the n-ary version of the &-rule has n premises containing only one of the component of the n-ary conjunction, and a copy of the context.*

▶ **Definition 43** (Processes-as-Formulas)**.** *The formula $[\![P]\!]$ associated to a process P is inductively defined as follows:*

$$\begin{array}{lll} [\![\mathsf{Nil}]\!] = \circ \quad [\![P \mid Q]\!] = [\![P]\!] \parr [\![Q]\!] & & [\![(\nu x)(P)]\!] = \text{И}x.[\![P]\!] \\ \qquad [\![x!\langle y\rangle.P]\!] = \langle x!y\rangle ◀ [\![P]\!] & & [\![x?(y).P]\!] = \exists y.((x?y) ◀ [\![P]\!]) \\ [\![x \triangleright \{\ell : P_\ell\}_{\ell\in L}]\!] = \bigoplus_{\ell\in L} ((x?\ell) ◀ [\![P_\ell]\!]) & & [\![x \triangleleft \{\ell : P_\ell\}_{\ell\in L}]\!] = \mathop{\&}_{\ell\in L} (\langle x!\ell\rangle ◀ [\![P_\ell]\!]) \end{array} \qquad (15)$$

We denote by $\lfloor\!\lfloor P \rfloor\!\rfloor$ the sequent obtained by recursively removing all top-level $\parr$-connectives and nominal quantifiers from the sequent, as well as units and unary additive connectives. If the obtained sequent is empty, then we let $\lfloor\!\lfloor P \rfloor\!\rfloor = \circ$.

▶ **Corollary 44.** *Let P and Q be processes. If $P \equiv Q$, then $[\![Q]\!] \multimapboth [\![P]\!]$.*

▶ **Notation 45.** *When translating the processes $x!\langle y\rangle = x!\langle y\rangle.\mathsf{Nil}$ (and similarly for $x?(y)$), we will simply write $[\![x!\langle y\rangle.\mathsf{Nil}]\!] = \langle x!y\rangle$ since $[\![x!\langle y\rangle.\mathsf{Nil}]\!] = \langle x!y\rangle \blacktriangleleft \circ$ is logically equivalent to $\langle x!y\rangle$.*

▶ Remark 46. For a counter-example of processes which are not $\equiv$-equivalent, but whose corresponding formulas are $\multimapboth$-equivalent, consider the process where fresh new name is chosen before a choice, and the one in which a fresh name is chosen after a choice.

$$P = (\nu x)\,(x \triangleleft \{\ell : P_\ell\}_{\ell\in L}) \qquad \text{and} \qquad Q = x \triangleleft \{\ell : (\nu x)P_\ell\}_{\ell\in L} \tag{16}$$

These processes are not equivalent modulo the structural equivalence $\equiv$ provided in literature [85, 32], which is the same as the one we provide in Figure 2.

It is worth noticing that Milner's original π-calculus includes the structural equivalence in the top of Equation (17), while, at the best of our knowledge, the literature on the π-calculus as presented in this paper (that is, as in [85, 32]) does not include the corresponding structural equivalence in the bottom of Equation (17), required to capture similar interactions between choices and restriction.

$$\frac{(\nu x)\,(A+B) \quad \equiv \quad (\nu x)A + (\nu x)B}{\begin{array}{ccc} x \triangleleft \{\ell : (\nu y)P_\ell\}_{\ell\in L} & \equiv & (\nu y)\,(x \triangleleft \{\ell : P_\ell\}_{\ell\in L}) \\ x \triangleright \{\ell : (\nu y)P_\ell\}_{\ell\in L} & \equiv & (\nu y)\,(x \triangleright \{\ell : P_\ell\}_{\ell\in L}) \end{array}} \tag{17}$$

This rises an interesting question on why those structural equivalences have not being used in the literature, even if the two processes in Equation (16) have the same behavior with respect to the results in session types.

As shown in detail in [8], it is possible to associate to each execution tree of a process P to an open derivation in PiL of a formula $[\![P]\!]$, therefore to characterize deadlock-freedom in terms of derivability in PiL. We report here only a sketch of the proof of this result.

▶ **Theorem 47** ([8]). *Let P be a process.*
1. *If P is a deadlock-free, then $\vdash_{\mathsf{PiL}} [\![P]\!]$.*
2. *If P is race-free, then P is deadlock-fre iff $\vdash_{\mathsf{PiL}} [\![P]\!]$.*

Sketch of proof. If P is deadlock-free, then each (maximal) execution tree $\mathcal{T}$ of P has leaves Nil. Since terms are considered up-to structural equivalence, we can assume without loss of generality that no child of a process P contains more occurrences of Nil than P. This can be obtained by orienting the structural equivalence $P \mid \mathsf{Nil} \equiv P$ in the natural way.

For each such tree, we define a derivation $[\![\mathcal{T}]\!]$ by induction on the structure of $\mathcal{T}$ as shown in Figure 17. Item 1 follows by definition. To prove Item 2, we show that we can transform a derivation of a formula $[\![P]\!]$ into a derivation made of blocks of rules as in Figure 17 using rule permutations from Figure 16. We conclude by remarking that it suffices to check a unique derivation because when P is deadlock-free, then all derivations of $[\![P]\!]$ are equivalent with respect to the interleaving relation defined in Figure 3. ◀

▶ Remark 48. Certain works on π-calculus (e.g., [80]) restrict communication and selection on restricted channels (i.e., communication or selection on a channel y can be performed only if y is bound by a ν). To capture such a restriction it would be sufficient to require that an ax-rule with conclusion $\langle x!y\rangle$ and $(x?y)$ can be applied only if the x is bounded by a И in a sequent occurring in the derivation below the rule. In the proof net defined in Section 5, this restriction corresponds to require that the two formulas in an axiomatic link $a = \{\ \langle x!y\rangle\ ,\ (v?w)\ \}$ contains two vertices above a same $\text{И}z$ node.

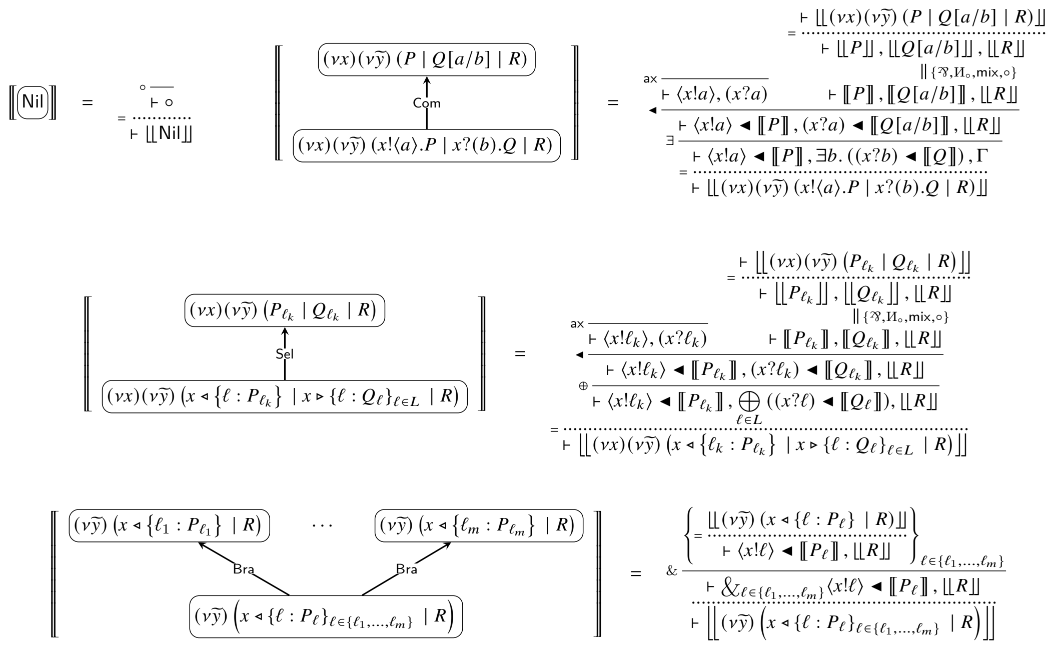


Figure 17 Translation of execution trees to derivations. If $m = 1$ in the case of Bra, then the open derivation is just the sequent $\left\lfloor\!\left\lfloor (\nu\tilde{y}) \left(x \triangleleft \{\ell : P_\ell\}_{\ell\in\{\ell_1,\ldots,\ell_m\}} \mid R\right)\right\rfloor\!\right\rfloor$, i.e., no rule.

The canonicity result on proof nets with respect to the sequent calculus allows us to provide canonical representatives of execution trees modulo inteleaving. To prove this result not only for deadlock-free processes, we extend the syntax of proof nets to include *non-logical axioms* to model open premises of a derivation.

▶ **Definition 49.** *A* ***open conflict net*** *(resp. open slice net) is a coalescent canonical co-tree $\tau(\Lambda)$ (resp. an erasable set of axiomatic linkings) on a sequent Γ. The (top-down) translation in Figure 11 is extended from derivations to open derivations by translating each open premise $\mathcal{S} \vdash A_1, \ldots, A_n$ in a the link $a = \{A_1, \ldots, A_n\}$ with $\delta_a = \emptyset$.*

▶ **Theorem 50.** *Two execution trees of a processes P are equivalent modulo interleaving (resp. local interleaving) iff they can be represented by the same open slice net (resp. open conflict net).*

Proof. We associate each execution tree $\mathcal{T}$ the slice net $\{\{\mathcal{T}\}\}_{\mathsf{sl}}$ by combining the translations in Figure 17 and Figure 15 (plus the special case for open premises defined in Definition 49). If $\mathcal{T} \sim \mathcal{T}'$, then $[\![\mathcal{T}]\!] \simeq [\![\mathcal{T}']\!]$; therefore $\{\{\mathcal{T}\}\}_{\mathsf{sl}} = \{\{\mathcal{T}'\}\}_{\mathsf{sl}}$ by Theorem 41. The result for local interleaving is similar. ◀

▶ **Corollary 51.** *A process P is race-free iff there is a unique possible slice net for $[\![P]\!]$.*

8 Conclusion and Future Works

In this paper we presented PiL, an extension of first-order multiplicative-additive linear logic in which non-commutativity is not obtained by considering sequents as lists of formulas (as in [1, 79, 31]), but rather including a non-commutative binary connective, as in Retoré's Pomset logic [75, 76] or Guglielmi's BV [40]. We have shown that by requiring such non-commutative connective to also be non-associative, we can design sequent calculus in which the cut-rule is admissible. We also provided proof nets for this logic with a polynomial-time correctness criterion (that is, proof nets form a proof system in the sense of [24]), polynomial-time sequentialization and proof translation, and we showed that proof nets provide canonical representatives of derivations modulo local rule permutations. Note that our result is stronger than what needed to define proof nets (both conflict and slice nets) for MALL^1, addressing a question left open in the literature. Moreover, it could be possible to define both conflict and slice 'witness nets' for PiL (and then for MALL^1) by including in the initial data of the proof net the information of a specific witness map instead of the initial witness map.

Characerization of deadlock-freedom. In [8] we have shown that each derivation in PiL of the formula $[\![P]\!]$ can be interpreted as a execution tree of the process P of the π-calculus (defined as in [85, 32]). In the same paper, we used such a correspondence to show that (finite) deadlock-freedom and race-free processes can be characterized in terms of derivability in PiL. To remove the race-free condition, we should be able to reason over all possible proof-search attempts, which has an exponential blowup because of interleaving. We are currently studying the possibility of reducing this blowup by using slice nets, allowing us to quantify over sets of linkings which may be valid slice nets instead of all possible "maximal open derivations".

Extensions of PiL with fixpoints. In this work, we have studied the recursion-free fragment of the π-calculus, but we foresee the possibility of modeling the following three main approaches for the definition of infinite behaviors (see [22] for a comparison of their expressive power). *Replication* (resp. *iteration*) could be modeled using the modality *why-not* (resp. *flag*) defined as fixpoint of the equation $?A = A \parr (?A)$ (resp. $\urcorner A = A \blacktriangleleft (\urcorner A)$) as in parsimonious linear logic [62, 4, 5] (resp. as a "parsimonious" version of the modalities from [74] and from [77]), and *recursion* using the greatest-fixpoint operator (which we here denote $\nu X.$ to avoid confusion with the restriction) from μMALL [17, 16].

Replication	Iteration	Recursion
$?\mathsf{b}\,\dfrac{\mathcal{S} \vdash \Gamma, A, ?A}{\mathcal{S} \vdash \Gamma, ?A}$	$\urcorner\mathsf{b}\,\dfrac{\mathcal{S} \vdash \Gamma, A \blacktriangleleft \urcorner A}{\mathcal{S} \vdash \Gamma, \urcorner A}$	$\nu X.\,\dfrac{\mathcal{S} \vdash \Gamma, P(\nu X.A)}{\mathcal{S} \vdash \Gamma, \nu X.A.P(A)}$

(18)

Proof systems capturing these operators should include rules allowing the definition of correct non-wellfounded derivations as in, e.g., [16, 4, 9]. An interesting challenge will be to find a suitable syntax for infinitary proof nets for these systems aiming at proof canonicity rather than to well-behavior with respect to cut-elimination as in [26, 25].

Coherent spaces for NML. As a consequence of Theorem 17 and cut-elimination, we have that NML embeds in BV, therefore NML embeds in Pomset. Since Pomset admits a semantics in terms of coherent spaces [75, 76], it should be possible to characterize the class of NML theorems inside Pomset, and use this logic to study sequential algorithms [74, 28, 29].

Applications to concurrent programming languages. This paper represents the first step in the definition of a novel paradigm allowing to intepret proofs as execution trees, and using proof equivalence to identify execution trees differing from interleaving concurrency, allowing for the developments of new methods complementary to the ones based session types.

In [8] we have shown that deadlock-freedom can be characterized in terms of derivability in PiL, providing the first completeness result for choreographic programming with respect to recursion-free deadlock-freedom. This was an open question in the literature, and the main difficulty of proving this result was due to syntactic limitations of session types in presence of key features such as name mobility or cyclic dependencies. By developing a theory of PiL with fixpoints, we expect to be able to extend this result to the full π-calculus.

We plan to study applications of our framework such as the definition of a notion of orthogonality for formulas based on the existence of (open) proof nets which could be completed by set of axioms connecting them (see the notion of module in [14, 7]). This could be used to characterize the *testing preorders* [27, 43, 19], thus designing verification tools whose efficiency relies on the low complexity of the proof net correctness criterion, combined with the fact that a single proof net can encode an exponential number of equivalent executions.

Moreover, our methodology is complementary to algebraic methods which, to the best of our knowledge, currently lack of methods to model name mobility and restrictions. This because these features are strongly tight to quantification, and modeling quantification has proven to be a complex task in algebraic settings: while Boolean and Heyting algebras provide straightforward algebraic models for classical and intuitionistic propositional logic respectively, the algebraic structures modeling first-order classical logic have only recently been studied (see [20]).

Towards a semantics for sequentiality. In this paper we used the non-associative connective $\blacktriangleleft$ to model a special form of sequentiality, the prefix operator, but we consider extending this work to sequent calculi using graphical connectives (in the sense of [3]) to model more complex pattern of interactions as done in [6], as well as study deep inference systems to recover associativity of the sequential operator – thus more similar to the systems studied in [21, 48, 45].

References

1 V.Michele Abrusci and Paul Ruet. Non-commutative logic I: the multiplicative fragment. *Annals of Pure and Applied Logic*, 101(1):29–64, 1999. URL: `https://www.sciencedirect.com/science/article/pii/S0168007299000147`, `doi:10.1016/S0168-0072(99)00014-7`.

2 Matteo Acclavio. Exponentially handsome proof nets and their normalization. *Electronic Proceedings in Theoretical Computer Science*, 353:1–25, Dec 2021. URL: `http://dx.doi.org/10.4204/EPTCS.353.1`, `doi:10.4204/eptcs.353.1`.

3 Matteo Acclavio. Sequent systems on undirected graphs. In Christoph Benzmüller, Marijn J.H. Heule, and Renate A. Schmidt, editors, *Automated Reasoning*, pages 216–236, Cham, 2024. Springer Nature Switzerland.

4 Matteo Acclavio, Gianluca Curzi, and Giulio Guerrieri. Infinitary cut-elimination via finite approximations. *CoRR*, abs/2308.07789, 2023. URL: `https://doi.org/10.48550/arXiv.2308.07789`, `arXiv:2308.07789`, `doi:10.48550/ARXIV.2308.07789`.

5 Matteo Acclavio, Gianluca Curzi, and Giulio Guerrieri. Infinitary cut-elimination via finite approximations (extended version), 2024. URL: `https://arxiv.org/abs/2308.07789`, `arXiv:2308.07789`.

6 Matteo Acclavio, Ross Horne, Sjouke Mauw, and Lutz Straßburger. A Graphical Proof Theory of Logical Time. In Amy P. Felty, editor, *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022)*, volume 228 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:25, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.FSCD.2022.22`, `doi:10.4230/LIPIcs.FSCD.2022.22`.

7 Matteo Acclavio and Roberto Maieli. Logic programming with multiplicative structures, 2024. URL: `https://arxiv.org/abs/2403.03032`, `arXiv:2403.03032`.

8 Matteo Acclavio, Giulia Manara, and Fabrizio Montesi. Formulas as processes, deadlock-freedom as choreographies (extended version), 2025. URL: `https://arxiv.org/abs/2501.08928`, `arXiv:2501.08928`.

9 Matteo Acclavio, Fabrizio Montesi, and Marco Peressotti. On propositional dynamic logic and concurrency, 2024. arXiv:2403.18508.

10 Matteo Acclavio and Lutz Straßburger. On combinatorial proofs for logics of relevance and entailment. In Rosalie Iemhoff, Michael Moortgat, and Ruy de Queiroz, editors, *Logic, Language, Information, and Computation*, pages 1–16, Berlin, Heidelberg, 2019. Springer Berlin Heidelberg.

11 Matteo Acclavio and Lutz Straßburger. On combinatorial proofs for modal logic. In Serenella Cerrito and Andrei Popescu, editors, *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 223–240, Cham, 2019. Springer International Publishing.

12 Alan Ross Anderson and Nuel D. jun. Belnap. Entailment. The logic of relevance and necessity. Vol. I. Princeton, N. J.: Princeton University Press. XXXII, 543 p. $ 23.50 (1975)., 1975.

13 Jean Marc Andreoli. Focussing proof-net construction as a middleware paradigm. In Andrei Voronkov, editor, *Automated Deduction—CADE-18*, pages 501–516, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

14 Jean-Marc Andreoli and Laurent Mazaré. Concurrent construction of proof-nets. In Matthias Baaz and Johann A. Makowsky, editors, *Computer Science Logic*, pages 29–42, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

15 Arnon Avron and Iddo Lev. Canonical propositional Gentzen-type systems. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Automated Reasoning*, pages 529–544, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

16 David Baelde, Amina Doumane, and Alexis Saurin. Infinitary proof theory: the multiplicative additive case. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 42:1–42:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CSL.2016.42.

17 David Baelde and Dale Miller. Least and greatest fixed points in linear logic. In Nachum Dershowitz and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference, LPAR 2007, Yerevan, Armenia, October 15-19, 2007, Proceedings*, volume 4790 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2007. doi:10.1007/978-3-540-75560-9_9.

18 Gianluigi Bellin. Subnets of proof-nets in multiplicative linear logic with mix. *Mathematical Structures in Computer Science*, 7(6):663–669, 1997.

19 Giovanni Bernardi and Matthew Hennessy. Mutually Testing Processes. *Logical Methods in Computer Science*, Volume 11, Issue 2, April 2015. URL: https://lmcs.episciences.org/776, doi:10.2168/LMCS-11(2:1)2015.

20 Filippo Bonchi, Alessandro Di Giorgio, Nathan Haydon, and Pawel Sobocinski. Diagrammatic algebra of first order logic. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '24, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3661814.3662078.

21 Paola Bruscoli. A purely logical account of sequentiality in proof search. In *International Conference on Logic Programming*, pages 302–316. Springer, 2002.

22 Nadia Busi, Maurizio Gabbrielli, and Gianluigi Zavattaro. On the expressive power of recursion, replication and iteration in process calculi. *Mathematical Structures in Computer Science*, 19(6):1191–1222, 2009. doi:10.1017/S096012950999017X.

23 James Cheney. A simpler proof theory for nominal logic. In Vladimiro Sassone, editor, *Foundations of Software Science and Computational Structures*, pages 379–394, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

24 Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979. doi:10.2307/2273702.

25 Abhishek De. *Linear logic with the least and greatest fixed points : truth semantics, complexity and a parallel syntax. (La logique linéaire avec les plus petits et les plus grands points fixes*

: la sémantique de vérité, la complexité, et une syntaxe parallèle). PhD thesis, Paris Cité University, France, 2022. URL: `https://tel.archives-ouvertes.fr/tel-04274800`.

26 Abhishek De and Alexis Saurin. Infinets: The parallel syntax for non-wellfounded proof-theory. In Serenella Cerrito and Andrei Popescu, editors, *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 297–316, Cham, 2019. Springer International Publishing.

27 Rocco De Nicola and Matthew CB Hennessy. Testing equivalences for processes. *Theoretical computer science*, 34(1-2):83–133, 1984.

28 Thomas Ehrhard. Projecting sequential algorithms on strongly stable functions. *Annals of Pure and Applied Logic*, 77(3):201–244, 1996. URL: `https://www.sciencedirect.com/science/article/pii/0168007295000267`, `doi:10.1016/0168-0072(95)00026-7`.

29 Thomas Ehrhard. Parallel and serial hypercoherences. *Theoretical Computer Science*, 247(1):39–81, 2000. URL: `https://www.sciencedirect.com/science/article/pii/S0304397500001730`, `doi:10.1016/S0304-3975(00)00173-0`.

30 Murdoch J. Gabbay and Andrew M. Pitts. A new approach to abstract syntax with variable binding. *Form. Asp. Comput.*, 13(3–5):341–363, jul 2002. `doi:10.1007/s001650200016`.

31 D. Galmiche and J. M. Notin. Connection-based proof construction in non-commutative logic. In Moshe Y. Vardi and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 422–436, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

32 Simon Gay and Malcolm Hole. Subtyping for session types in the pi calculus. *Acta Informatica*, 42:191–225, 2005.

33 Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987. `doi:10.1016/0304-3975(87)90045-4`.

34 Jean-Yves Girard. Quantifiers in linear logic. In *Proceedings of the SILFS conference held in Cesena*, volume 41, 1987.

35 Jean-Yves Girard. Quantifiers in linear logic ii. *Nuovi problemi della logica e della filosofia della scienza*, 2(1), 1991.

36 Jean-Yves Girard. Proof-nets : the parallel syntax for proof-theory. In Aldo Ursini and Paolo Agliano, editors, *Logic and Algebra*. Marcel Dekker, New York, 1996.

37 Jean-Yves Girard. The blind spot. *HAL*, 2011(0), 2011. URL: `http://dml.mathdoc.fr/item/ISBN:978-3-03719-088-3`.

38 Alessio Guglielmi. Concurrency and plan generation in a logic programming language with a sequential operator. In *ICLP*, pages 240–254. Citeseer, 1994.

39 Alessio Guglielmi. Sequentiality by linear implication and universal quantification. In Jörg Desel, editor, *Structures in Concurrency Theory*, pages 160–174, London, 1995. Springer London.

40 Alessio Guglielmi. A system of interaction and structure. 8(1):1–64, 2007. `doi:10.1145/1182613.1182614`.

41 Willem Heijltjes and Robin Houston. No proof nets for MLL with units: proof equivalence in MLL is PSPACE-complete. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 50:1–50:10. ACM, 2014.

42 Willem B. Heijltjes, Dominic J. D. Hughes, and Lutz Straßburger. Proof Nets for First-Order Additive Linear Logic. In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, volume 131 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:22, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.FSCD.2019.22`, `doi:10.4230/LIPIcs.FSCD.2019.22`.

43 Matthew Hennessy. *Algebraic theory of processes*. MIT press, 1988.

44 Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. *SIGPLAN Not.*, 43(1):273–284, jan 2008. `doi:10.1145/1328897.1328472`.

45 Ross Horne and Alwen Tiu. Constructing weak simulations from linear implications for processes with private names. *Mathematical Structures in Computer Science*, 29(8):1275–1308, 2019. doi:10.1017/S0960129518000452.

46 Ross Horne, Alwen Tiu, Bogdan Aman, and Gabriel Ciobanu. Private Names in Non-Commutative Logic. In Josée Desharnais and Radha Jagadeesan, editors, *27th International Conference on Concurrency Theory (CONCUR 2016)*, volume 59 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CONCUR.2016.31, doi:10.4230/LIPIcs.CONCUR.2016.31.

47 Ross Horne, Alwen Tiu, Bogdan Aman, and Gabriel Ciobanu. Private Names in Non-Commutative Logic. In Josée Desharnais and Radha Jagadeesan, editors, *27th International Conference on Concurrency Theory (CONCUR 2016)*, volume 59 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CONCUR.2016.31, doi:10.4230/LIPIcs.CONCUR.2016.31.

48 Ross Horne, Alwen Tiu, Bogdan Aman, and Gabriel Ciobanu. De morgan dual nominal quantifiers modelling private names in non-commutative logic. *ACM Trans. Comput. Logic*, 20(4), jul 2019. doi:10.1145/3325821.

49 Dominic Hughes. Proofs Without Syntax. *Annals of Mathematics*, 164(3):1065–1076, 2006. doi:10.4007/annals.2006.164.1065.

50 Dominic Hughes. Towards Hilbert's 24^{th} problem: Combinatorial proof invariants: (preliminary version). *Electr. Notes Theor. Comput. Sci.*, 165:37–63, 2006.

51 Dominic Hughes. First-order proofs without syntax. Berkeley Logic Colloquium, 2014.

52 Dominic Hughes and Willem Heijltjes. Conflict nets: Efficient locally canonical mall proof nets. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '16, page 437–446, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2933575.2934559.

53 Dominic J. D. Hughes. Abstract p-time proof nets for mall: Conflict nets, 2008. URL: https://arxiv.org/abs/0801.2421, arXiv:0801.2421.

54 Dominic J. D. Hughes. Unification nets: canonical proof net quantifiers. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, page 540–549, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3209108.3209159.

55 Dominic J. D. Hughes and Rob J. Van Glabbeek. Proof nets for unit-free multiplicative-additive linear logic (extended abstract). In *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science*, LICS '03, page 1, USA, 2003. IEEE Computer Society.

56 Dominic J. D. Hughes and Rob J. Van Glabbeek. Proof nets for unit-free multiplicative-additive linear logic. *ACM Trans. Comput. Logic*, 6(4):784–842, oct 2005. doi:10.1145/1094622.1094629.

57 Andrzej Indrzejczak. Sequent calculi for monotonic modal logics. *Bulletin of the Section of Logic*, 34(3):151–164, 2005.

58 Joachim Lambek. Deductive systems and categories I, II, III, 1968-1972.

59 Björn Lellmann and Elaine Pimentel. Modularisation of sequent calculi for normal and non-normal modalities. *ACM Trans. Comput. Logic*, 20(2), feb 2019. doi:10.1145/3288757.

60 Giulia Manara. *Linear Logic: Parallel cut elimination and Computation-as-Deduction for the π-calculus.* Ph.D. thesis, Paris Cité University, France & University Roma Tre, Italy, 2025.

61 Alberto Martelli and Ugo Montanari. *Unification in linear time and space: A structured presentation.* Istituto di Elaborazione della Informazione, Consiglio Nazionale delle Ricerche, 1976.

62 Damiano Mazza. Simple parsimonious types and logarithmic space. In *24th EACSL Annual Conference on Computer Science Logic, CSL 2015*, volume 41 of *LIPIcs*, pages 24–40. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CSL.2015.24.

63 Matias Menni. About И-quantifiers. *Applied categorical structures*, 11:421–445, 2003.

64 Dale Miller. The π-calculus as a theory in linear logic: Preliminary results. In E. Lamma and P. Mello, editors, *Extensions of Logic Programming*, pages 242–264, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

65 Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51(1):125–157, 1991. URL: https://www.sciencedirect.com/science/article/pii/016800729190068W, doi:10.1016/0168-0072(91)90068-W.

66 Dale Miller and Elaine Pimentel. A formal framework for specifying sequent calculus proof systems. *Theoretical Computer Science*, 474:98–116, 2013.

67 Dale Miller and Alwen Tiu. A proof theory for generic judgments. *ACM Trans. Comput. Logic*, 6(4):749–783, oct 2005. doi:10.1145/1094622.1094628.

68 Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980. doi:10.1007/3-540-10235-3.

69 Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, i. *Information and Computation*, 100(1):1–40, 1992. URL: https://www.sciencedirect.com/science/article/pii/0890540192900084, doi:10.1016/0890-5401(92)90008-4.

70 Fabrizio Montesi. *Introduction to Choreographies*. Cambridge University Press, 2023. doi:10.1017/9781108981491.

71 Lê Thành Dũng Nguyên and Lutz Straßburger. A System of Interaction and Structure III: The Complexity of BV and Pomset Logic. working paper or preprint, 2022. URL: https://hal.inria.fr/hal-03909547.

72 Lê Thành Dũng Nguyên and Lutz Straßburger. BV and Pomset Logic are not the same. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic (CSL 2022)*, volume 216 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:17, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2022/15723, doi:10.4230/LIPIcs.CSL.2022.3.

73 Andrew M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186(2):165–193, 2003. Theoretical Aspects of Computer Software (TACS 2001). URL: https://www.sciencedirect.com/science/article/pii/S089054010300138X, doi:10.1016/S0890-5401(03)00138-X.

74 Uday S. Reddy. A linear logic model of state. 1993. URL: https://api.semanticscholar.org/CorpusID:10342053.

75 Christian Retoré. *Réseaux et Séquents Ordonnés*. PhD thesis, Université Paris VII, 1993.

76 Christian Retoré. Pomset logic: The other approach to noncommutativity in logic. *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics*, pages 299–345, 2021.

77 Christian Retoré. Flag: a self-dual modality for non-commutative contraction and duplication in the category of coherence spaces. *Electronic Proceedings in Theoretical Computer Science*, 353:157–174, December 2021. URL: http://dx.doi.org/10.4204/EPTCS.353.8, doi:10.4204/eptcs.353.8.

78 Luca Roversi. A deep inference system with a self-dual binder which is complete for linear lambda calculus. *Journal of Logic and Computation*, 26(2):677–698, 2016. doi:10.1093/logcom/exu033.

79 Paul Ruet. Non-commutative logic II: sequent calculus and phase semantics. *Mathematical Structures in Computer Science*, 10(2):277–312, 2000. doi:10.1017/S0960129599003084.

80 Davide Sangiorgi. π-calculus, internal mobility, and agent-passing calculi. *Theoretical Computer Science*, 167(1):235–274, 1996. URL: https://www.sciencedirect.com/science/article/pii/0304397596000758, doi:10.1016/0304-3975(96)00075-8.

81 Sergey Slavnov. On noncommutative extensions of linear logic. *Logical Methods in Computer Science*, Volume 15, Issue 3, September 2019. URL: https://lmcs.episciences.org/5774, doi:10.23638/LMCS-15(3:30)2019.

82 Lutz Straßburger. The problem of proof identity, and why computer scientists should care about hilbert's 24th problem. *Philosophical Transactions of the Royal Society A*, 377(2140):20180038, 2019. doi:10.1098/rsta.2018.0038.

83 Alwen Fernanto Tiu. A system of interaction and structure II: The need for deep inference. 2(2):1–24, 2006. doi:10.2168/LMCS-2(2:4)2006.

84 A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2 edition, 2000. doi:10.1017/CBO9781139168717.

85 Vasco T. Vasconcelos. Fundamentals of session types. *Information and Computation*, 217:52–70, 2012. URL: https://www.sciencedirect.com/science/article/pii/S0890540112001022, doi:10.1016/j.ic.2012.05.002.

86 Glynn Winskel. Event structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, pages 325–392, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

$$
\begin{array}{rcll}
\mathsf{Nil} & =_\alpha & \mathsf{Nil} & \\
x?(y).P & =_\alpha & x?(z).P[z/y] & z \text{ fresh for } P \\
x!\langle y\rangle.P & =_\alpha & x!\langle y\rangle.Q & \text{if } P =_\alpha Q \\
P \mid Q & =_\alpha & R \mid S & \text{if } P =_\alpha R \text{ and } Q =_\alpha S \\
(\nu x)P & =_\alpha & (\nu u)P[u/x] & u \text{ fresh for } P \\
x \lhd \{\ell : P_\ell\}_{\ell \in L} & =_\alpha & x \lhd \{\ell : Q_\ell\}_{\ell \in L} & \text{if } P_\ell =_\alpha Q_\ell \text{ for all } \ell \in L \\
x \rhd \{\ell : P_\ell\}_{\ell \in L} & =_\alpha & x \rhd \{\ell : Q_\ell\}_{\ell \in L} & \text{if } P_\ell =_\alpha Q_\ell \text{ for all } \ell \in L
\end{array}
$$

Figure 18 Definition of $=_\alpha$ for processes.

MAV[1]-Formulas

$$A, B := \circ \mid A \parr B \mid A \lhd B \mid A \otimes B \mid A \oplus B \mid A \,\&\, B \mid ?_x A \mid !_x A \mid \forall x.A \mid \text{И}x.A \mid \text{Э}x.A \mid \exists x.A$$

Formula equivalences

$$A = A \parr \circ = A \lhd \circ = \circ \lhd A = A \otimes \circ$$
$$\circ = \circ \oplus \circ = \circ \,\&\, \circ = \forall x.\circ = \text{И}x.\circ = \text{Э}x.\circ = \exists x.\circ$$
$$A \odot B = B \odot A \text{ with } \odot \in \{\parr, \otimes, \oplus, \&\}$$
$$A \odot (B \odot C) = (A \odot B) \odot C \text{ with } \odot \in \{\parr, \otimes, \oplus, \&\}$$
$$A \lhd (B \lhd C) = (A \lhd B) \lhd C$$

Derivations

$$\mathcal{D} := \quad A \quad \Bigg| \quad \begin{array}{c} A \\ \mathcal{D}_1 \| \\ \mathsf{r}\,\dfrac{C'}{C} \\ \mathcal{D}_2 \| \\ B \end{array} \quad \Bigg| \quad \begin{array}{c} A_1 \\ \mathcal{D}_1 \| \\ B_1 \end{array} \odot \begin{array}{c} A_1 \\ \mathcal{D}_1 \| \\ B_1 \end{array} \quad \Bigg| \quad \mho x. \begin{array}{c} A_1 \\ \mathcal{D}_1 \| \\ B_1 \end{array} \quad \text{where} \quad \begin{array}{c} \mathcal{D}_1 \| \\ A \end{array} := \begin{array}{c} \circ \\ \mathcal{D}_1 \| \\ A \end{array}$$

with A, B, C, C' formulasand r rule and $\odot \in \{\parr, \otimes, \lhd, \oplus, \&\}$ and $\mho \in \{\exists, \forall, \text{И}, \text{Э}\}$

Rules

$$\mathsf{ai}{\downarrow}\,\frac{\circ}{a \parr a^\perp} \qquad \mathsf{s}\,\frac{A \otimes (B \parr C)}{(A \otimes B) \parr C} \qquad \mathsf{q}{\downarrow}\,\frac{(A \parr B) \lhd (C \parr D)}{(A \lhd C) \parr (B \lhd D)}$$

$$\&\,\frac{(A \parr B) \,\&\, (A \parr C)}{A \parr (B \,\&\, C)} \qquad \oplus\,\frac{A}{A \oplus B} \qquad \lhd\text{-}\&\,\frac{(A \,\&\, B) \lhd (C \,\&\, D)}{(A \lhd C) \,\&\, (B \lhd D)}$$

$$\mathsf{scope}_\forall\,\frac{\forall x.(A \parr F)}{(\forall x.A) \parr F} \qquad \lhd\text{-}\forall\,\frac{(\forall x.A) \lhd (\forall x.B)}{\forall x.(A \lhd B)} \qquad \exists\,\frac{A[c/x]}{\exists x.A}$$

$$\mathsf{scope}_\mho\,\frac{\mho x.(A \parr F)}{(\mho x.A) \parr F} \qquad \mathsf{scope}^{\lhd}_{\text{Э}}\,\frac{\text{Э}x.(A \lhd F)}{\text{Э}x.A \lhd F} \qquad \mathsf{scope}^{\lhd}_{\text{Э}}\,\frac{\text{Э}x.(F \lhd A)}{F \lhd \text{Э}x.A}$$

$$\mathsf{shift}\,\frac{\text{Э}x.\text{И}y.A}{\text{И}x.\text{Э}x.A} \qquad \text{И-Э}\,\frac{\text{И}x.(A \parr B)}{(\text{И}x.A) \parr (\text{Э}x.B)} \qquad \text{Э}\,\frac{\text{И}x.A}{\text{Э}x.A}$$

$$\mathsf{shift}_\forall\,\frac{\mho x.\forall y.A}{\forall x.\mho x.A} \qquad \mathsf{scope}^{\&}_{\mho}\,\frac{\mho x.(A \,\&\, F)}{(\mho x.A) \,\&\, F} \qquad \mathsf{nom{-}choice}\,\frac{\mho x.(A \,\&\, B)}{\mho x.A \,\&\, \mho x.B}$$

$$\lhd\text{-И}\,\frac{(\text{И}x.A) \lhd (\text{И}x.B)}{\text{И}x.(A \lhd B)} \qquad \text{Э-}\lhd\,\frac{\text{Э}x.(A \lhd B)}{\text{Э}x.A \lhd \text{Э}x.B}$$

where $\mho \in \{\text{И}, \text{Э}\}$ and $\odot \in \{\parr, \lhd\}$ and $x \notin \mathsf{free}(F)$

Systems

$\mathsf{BV} = \{\mathsf{ai}{\downarrow}, \mathsf{s}, \mathsf{q}{\downarrow}\}$ $\qquad$ $\mathsf{MAV} = \mathsf{BV} \cup \{\oplus, \&, \lhd\text{-}\&\}$

$\mathsf{BV}^1 = \mathsf{MAV}^1 \setminus \{\lhd\text{-И}, \text{Э-}\lhd\}$ $\qquad$ MAV^1 = all rules above

Figure 19 Inductive definition of deep inference derivation and the rules in the system MAV^1.

A Embedding PiL into MAV^1

We recall in Figure 19 the definition of MAV^1***-formulas***, formula equivalence, and deep inference derivations for MAV^1. Rules have been reorganized, also relying on a stronger formula equivalence capturing derivable equivalences involving additive connectives, to improve readability over the intuitive reading of the formula-as-process interpretation.

▶ **Theorem 17.** *Let $A_1, \ldots, A_n$ be formulas. If $\vdash_{\mathsf{PiL}} A_1, \ldots, A_n$, then $\vdash_{\mathsf{MAV}^1} \parr_{i=1}^n \lceil A_i \rceil$.*

Proof. For each derivation $\mathcal{D}$ in PiL conclusion $A_1, \ldots, A_n$, we define a deep-inference derivation $\lceil \mathcal{D} \rceil$ in MAV^1 with premise $\circ$ and conclusion $\parr_{i=1}^n \lceil A_i \rceil$ as shown in Figure 20. ◀

▶ Remark 52. In NML we have the same connectives $\parr$ and $\otimes$ of BV, as well as a non-commutative self-dual connective $\blacktriangleleft$, all sharing the same unit $\circ$. Moreover, as in BV, the implications $(A \otimes B) \multimap (A \blacktriangleleft B)$ and $(A \blacktriangleleft B) \multimap (A \parr B)$ hold, as well as the ones proving that $\circ$ is the unit for the three connectives $\parr$, $\blacktriangleleft$, and $\otimes$.

However, we know from [83] that BV cannot have cut-free a sequent calculus, and the same holds for Retore's Pomset[11] which is a proper conservative extension of BV [72, 71].

[11] Note that a cut-free sequent calculus for Pomset has been proposed in [81], but the side conditions of its sequent rules cannot be checked in polynomial time. Therefore such a sequent system cannot be considered a proper proof system, as intended in [24].

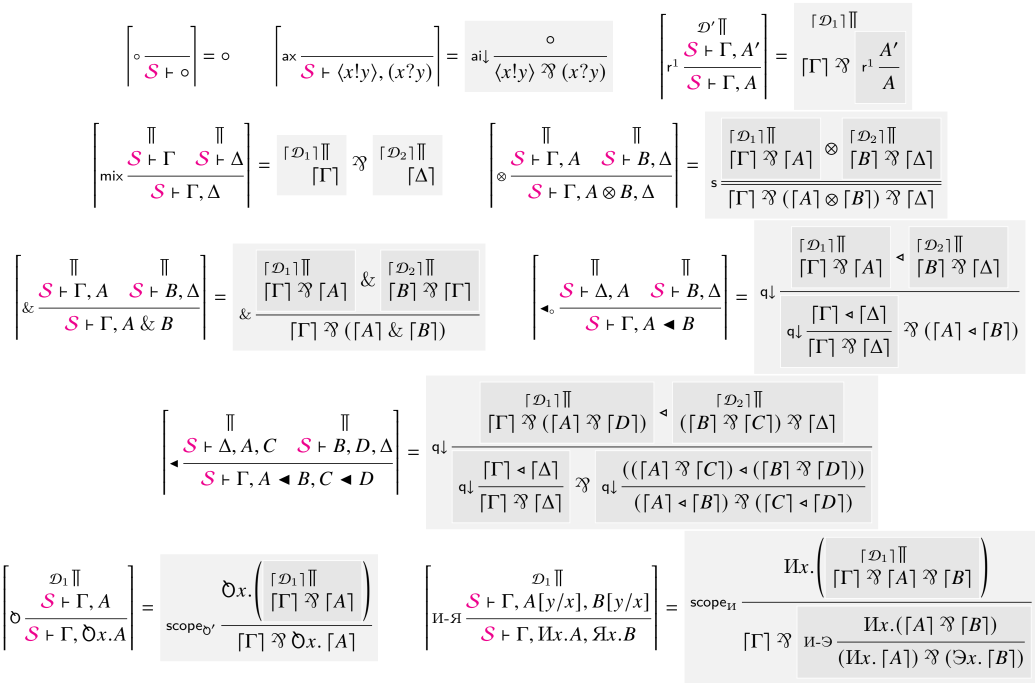


Figure 20 How to define a derivation $\lceil \mathcal{D} \rceil$ in MAV^1 from a derivation $\mathcal{D}$ in PiL, with $\mathsf{r}^1 \in \{⅋, \oplus, \exists\}$ and $\mho \in \{\forall, И, Я\}$ and $\mho' = \mho$ except if $\mho = Я$, in which case $\mho' = Э$.

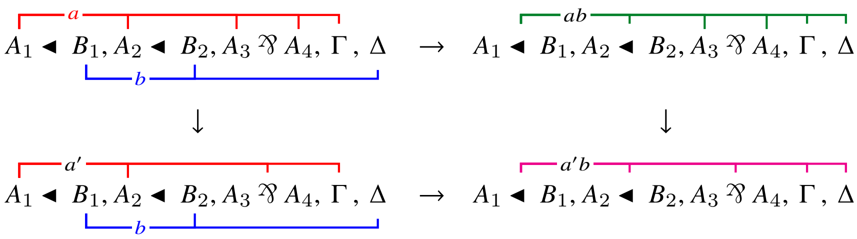
Thus we conjecture that the cause of the impossibility of having a cut-free sequent calculus for Guglielmi's BV [40] in the associativity of the connective $\triangleleft$.

B Confluence of coalescence

▶ **Lemma 40.** *Let $\tau(\Lambda)$ be a conflict net on Γ such that $\tau(\Lambda)$ sequetializes to $\mathcal{D}_{\tau(\Lambda)}$. If $\tau(\Lambda) \to^* \tau(\Lambda)'$, then $\tau(\Lambda)'$ sequentializes to a $\mathcal{D}_{\tau(\Lambda)'}$ such that $\mathcal{D}_{\tau(\Lambda)'} \approx_{\mathsf{w}} \mathcal{D}_{\tau(\Lambda)}$.*

Proof. We only discuss the critical pairs for coalescence rules not already discussed in [52]. Together with the confluence diagram of each critical pair, we show the two derivations corresponding to the two sequences of coalescence steps.

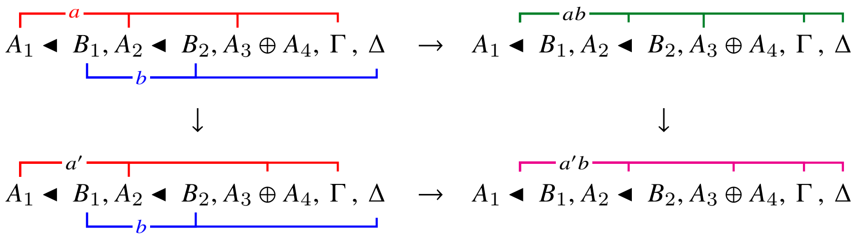
- Case $⅋/\blacktriangleleft$:



With $\delta_{a'} = \delta_a$ and $\delta_{ab} = \delta_{a'b} = \delta_a + \delta_b$.
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\dfrac{\dfrac{\mathcal{S} \vdash A_1, A_2, A_3 \parr A_4, \Gamma \quad \mathcal{S} \vdash B_1, B_2, \Delta}{\mathcal{S} \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, A_3, A_4, \Gamma, \Delta}\,\blacktriangleleft}{\mathcal{S} \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, A_3 \parr A_4, \Gamma, \Delta}\,\parr \quad \approx \quad \dfrac{\dfrac{\mathcal{S} \vdash A_1, A_2, A_3, A_4, \Gamma}{\mathcal{S} \vdash A_1, A_2, A_3 \parr A_4, \Gamma}\,\parr \quad \mathcal{S} \vdash B_1, B_2, \Delta}{\mathcal{S} \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, A_3 \parr A_4, \Gamma, \Delta}\,\blacktriangleleft$$

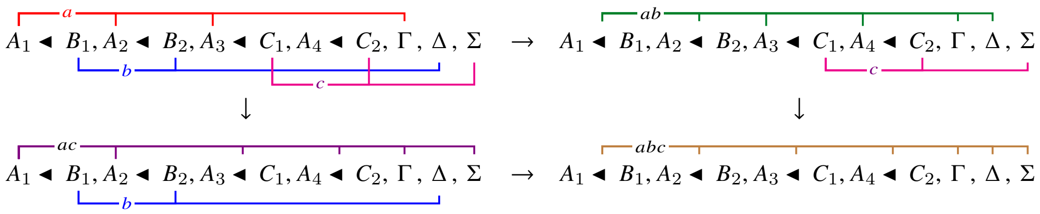
- Case $\oplus/\blacktriangleleft$:



With $\delta_{a'} = \delta_a$ and $\delta_{ab} = \delta_{a'b} = \delta_a + \delta_b$.
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\dfrac{\dfrac{\mathcal{S} \vdash A_1, A_2, A_3, \Gamma \quad \mathcal{S} \vdash B_1, B_2, \Delta}{\mathcal{S} \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, A_3, \Gamma, \Delta}\,\blacktriangleleft}{\mathcal{S} \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, A_3 \oplus A_4, \Gamma, \Delta}\,\oplus \quad \approx \quad \dfrac{\dfrac{\mathcal{S} \vdash A_1, A_2, A_3, \Gamma}{\mathcal{S} \vdash A_1, A_2, A_3, \Gamma}\,\oplus \quad \mathcal{S} \vdash B_1, B_2, \Delta}{\mathcal{S} \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, A_3 \oplus A_4, \Gamma, \Delta}\,\blacktriangleleft$$

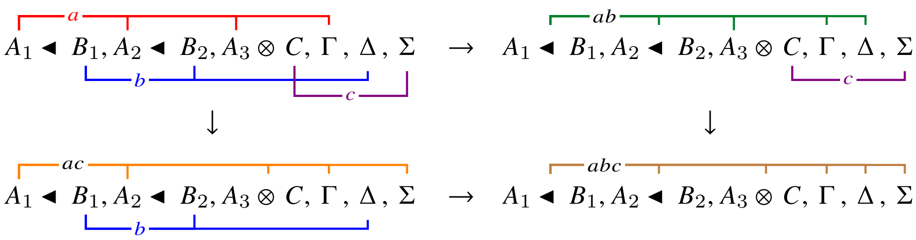
- Case $\blacktriangleleft/\blacktriangleleft$:



With $\delta_{ab} = \delta_a + \delta_b$, $\delta_{ac} = \delta_a + \delta_c$ and $\delta_{abc} = \delta_a + \delta_b + \delta_c$.
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\dfrac{\dfrac{\mathcal{S} \vdash A_1, A_2, A_3, A_4, \Gamma \quad \mathcal{S} \vdash C_1, C_2, \Sigma}{\mathcal{S} \vdash A_1, A_2, A_3 \blacktriangleleft C_1, A_4 \blacktriangleleft C_2, \Gamma, \Sigma}\,\blacktriangleleft \quad \mathcal{S} \vdash B_1, B_2, \Delta}{\mathcal{S} \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, A_3 \blacktriangleleft C_1, A_4 \blacktriangleleft C_2, \Gamma, \Delta, \Sigma}\,\blacktriangleleft \quad \approx \quad \dfrac{\dfrac{\mathcal{S} \vdash A_1, A_2, A_3, A_4, \Gamma \quad \mathcal{S} \vdash B_1, B_2, \Delta}{\mathcal{S} \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, A_3, A_4, \Gamma, \Delta}\,\blacktriangleleft \quad \mathcal{S} \vdash C_1, C_2, \Sigma}{\mathcal{S} \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, A_3 \blacktriangleleft C_1, A_4 \blacktriangleleft C_2, \Gamma, \Delta, \Sigma}\,\blacktriangleleft$$

- Case $\otimes/\blacktriangleleft$:

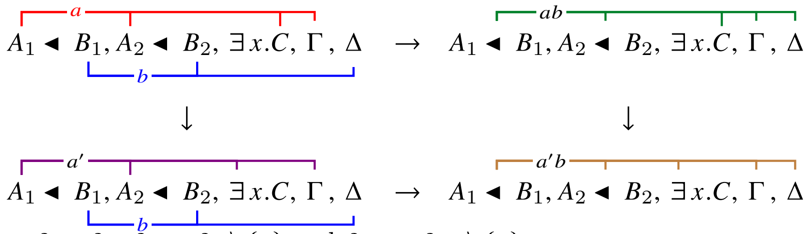


With $\delta_{ab} = \delta_a + \delta_b$, $\delta_{ac} = \delta_a + \delta_c$ and $\delta_{abc} = \delta_a + \delta_b + \delta_c$.
The two distinct derivations labeling the link in the bottom-right corner of the diagram

according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\dfrac{\dfrac{\mathcal{S}\vdash A_1,A_2,A_3,A_4,\Gamma \quad \mathcal{S}\vdash B_1,B_2,\Delta}{\mathcal{S}\vdash A_1,A_2,A_3\otimes C,\Gamma,\Sigma}{\scriptstyle\blacktriangleleft} \qquad \mathcal{S}\vdash C,\Sigma}{\mathcal{S}\vdash A_1\blacktriangleleft B_1,A_2\blacktriangleleft B_2,A_3\otimes C,\Gamma,\Delta,\Sigma}{\scriptstyle\otimes} \approx \dfrac{\dfrac{\mathcal{S}\vdash A_1,A_2,A_3,A_4,\Gamma \quad \mathcal{S}\vdash C,\Sigma}{\mathcal{S}\vdash A_1\blacktriangleleft B_1,A_2\blacktriangleleft B_2,A_3,A_4,\Gamma,\Delta}{\scriptstyle\otimes} \qquad \mathcal{S}\vdash B_1,B_2,\Delta}{\mathcal{S}\vdash A_1\blacktriangleleft B_1,A_2\blacktriangleleft B_2,A_3\otimes C,\Gamma,\Delta,\Sigma}{\scriptstyle\blacktriangleleft}$$

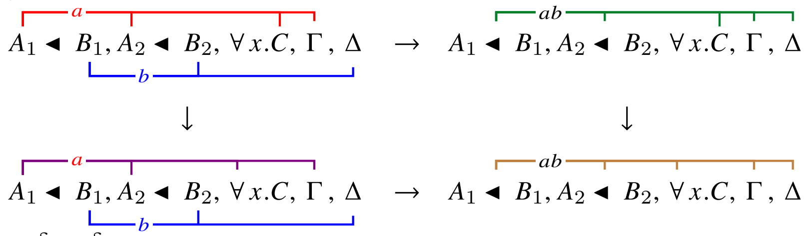
- Case $\exists/\blacktriangleleft$:



With $\delta_{ab}=\delta_a+\delta_b$, $\delta_{a'}=\delta_a\setminus\{x\}$ and $\delta_{a'b}=\delta_{ab}\setminus\{x\}$
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\dfrac{\dfrac{\mathcal{S}\vdash A_1,A_2,C[c/x],\Gamma \quad \mathcal{S}\vdash B_1,B_2,\Delta}{\mathcal{S}\vdash A_1\blacktriangleleft B_1,A_2\blacktriangleleft B_2,C[c/x],\Gamma,\Delta}{\scriptstyle\blacktriangleleft}}{\mathcal{S}\vdash A_1\blacktriangleleft B_1,A_2\blacktriangleleft B_2,\exists xC,\Gamma,\Delta}{\scriptstyle\exists} \approx \dfrac{\dfrac{\mathcal{S}\vdash A_1,A_2,C[c/x],\Gamma}{\mathcal{S}\vdash A_1,A_2,\exists xC,\Gamma}{\scriptstyle\exists} \qquad \mathcal{S}\vdash B_1,B_2,\Delta}{\mathcal{S}\vdash A_1\blacktriangleleft B_1,A_2\blacktriangleleft B_2,\exists xC,\Gamma,\Delta}{\scriptstyle\blacktriangleleft}$$

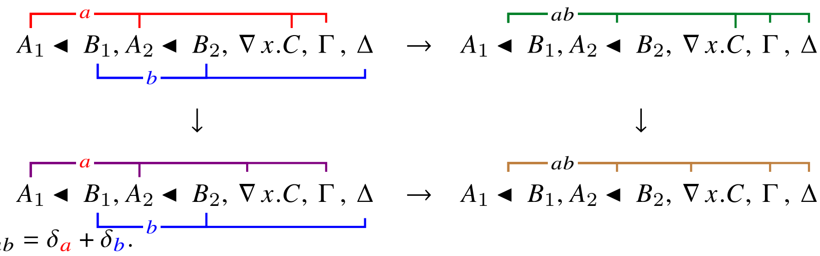
- Case $\forall/\blacktriangleleft$:



With $\delta_{ab}=\delta_a+\delta_b$.
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\dfrac{\dfrac{\mathcal{S}\vdash A_1,A_2,C,\Gamma \quad \mathcal{S}\vdash B_1,B_2,\Delta}{\mathcal{S}\vdash A_1\blacktriangleleft B_1,A_2\blacktriangleleft B_2,C,\Gamma,\Delta}{\scriptstyle\blacktriangleleft}}{\mathcal{S}\vdash A_1\blacktriangleleft B_1,A_2\blacktriangleleft B_2,\forall x.C,\Gamma,\Delta}{\scriptstyle\forall} \approx \dfrac{\dfrac{\mathcal{S}\vdash A_1,A_2,C,\Gamma}{\mathcal{S}\vdash A_1,A_2,\forall x.C,\Gamma}{\scriptstyle\forall} \qquad \mathcal{S}\vdash B_1,B_2,\Delta}{\mathcal{S}\vdash A_1\blacktriangleleft B_1,A_2\blacktriangleleft B_2,\forall x.C,\Gamma,\Delta}{\scriptstyle\blacktriangleleft}$$

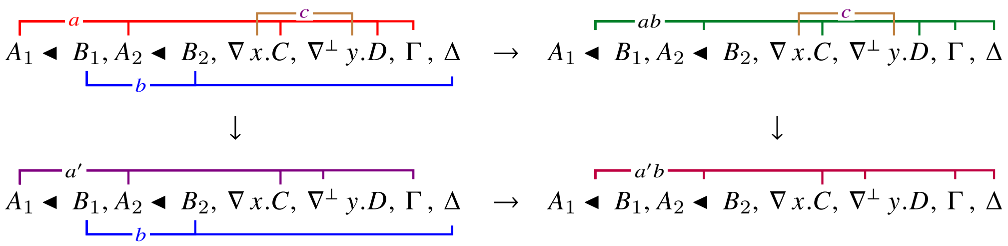
- Case $\nabla_{\mathsf{load}}/\blacktriangleleft$ with $\nabla\in\{\text{И},\text{Я}\}$: similarly to the case $\forall/\blacktriangleleft$, but considering the rule ∇_{load} and the nominal quantifier ∇ instead of $\forall$.



With $\delta_{ab}=\delta_a+\delta_b$.
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\dfrac{\dfrac{x\vdash A_1,A_2,C,\Gamma \quad x\vdash B_1,B_2,\Delta}{x\vdash A_1\blacktriangleleft B_1,A_2\blacktriangleleft B_2,C,\Gamma,\Delta}{\scriptstyle\blacktriangleleft}}{\mathcal{S}\vdash A_1\blacktriangleleft B_1,A_2\blacktriangleleft B_2,\nabla x.C,\Gamma,\Delta}{\scriptstyle\nabla_{\mathsf{load}}} \approx \dfrac{\dfrac{x\vdash A_1,A_2,C,\Gamma}{\mathcal{S}\vdash A_1,A_2,\nabla x.C,\Gamma}{\scriptstyle\nabla_{\mathsf{load}}} \qquad \mathcal{S}\vdash B_1,B_2,\Delta}{\mathcal{S}\vdash A_1\blacktriangleleft B_1,A_2\blacktriangleleft B_2,\nabla x.C,\Gamma,\Delta}{\scriptstyle\blacktriangleleft}$$

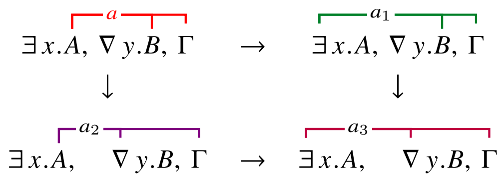
- Case $\nabla_{\circ}/\blacktriangleleft$ with $\nabla \in \{\text{И}, \text{Я}\}$: similar to the previous case, but considering the rule $\nabla_{\circ}$ instead of ∇_{load}.
- Case $\nabla_{\mathsf{pop}}/\blacktriangleleft$ with $\nabla \in \{\text{И}, \text{Я}\}$:



With $\delta_{a'} = \delta_a \setminus \{y\}$, $\delta_{ab} = \delta_a + \delta_b$ and $\delta_{a'b} = \delta_{ab} \setminus \{y\}$.
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\nabla_{\mathsf{pop}}\dfrac{\blacktriangleleft\dfrac{\mathcal{S}_1 \vdash A_1, A_2, C, D[x/y], \Gamma \quad \mathcal{S}_1 \vdash B_1, B_2, \Delta}{\mathcal{S}_1, \mathcal{S}_2 \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, C, D[x/y], \Gamma, \Delta}}{\mathcal{S}_1, \mathcal{S}_2, x^{\nabla} \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, C, \nabla^{\perp} y.D, \Gamma, \Delta} \approx \blacktriangleleft\dfrac{\nabla_{\mathsf{pop}}\dfrac{\mathcal{S}_1 \vdash A_1, A_2, C, D[x/y], \Gamma}{\mathcal{S}_1, x^{\nabla} \vdash A_1, A_2, C, \nabla^{\perp} y.D, \Gamma} \quad \mathcal{S}_2 \vdash B_1, B_2, \Delta}{\mathcal{S}_1, x^{\nabla} \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, C, \nabla^{\perp} y.D, \Gamma, \Delta}$$

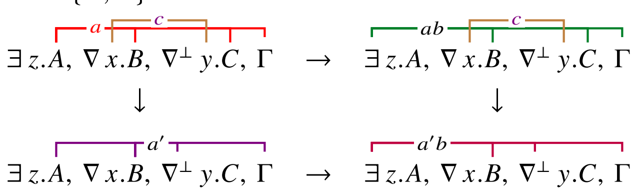
- Case $\exists/\nabla_{\mathsf{load}}$ with $\nabla \in \{\text{И}, \text{Я}\}$:



With $\delta_{a_1} = \delta_a \setminus \{x\}$, $\delta_{a_2} = \delta_a$ and $\delta_{a_3} = \delta_{a_1}$.
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\nabla_{\mathsf{load}}\dfrac{\exists\dfrac{\mathcal{S}, y^{\nabla} \vdash A[c/x], B, \Gamma}{\mathcal{S}, y^{\nabla} \vdash \exists x.A, B, \Gamma}}{\mathcal{S} \vdash \exists x.A, \nabla y.B, \Gamma} \approx \exists\dfrac{\nabla_{\mathsf{load}}\dfrac{\mathcal{S}, y^{\nabla} \vdash A[c/x], B, \Gamma}{\mathcal{S} \vdash A[c/x], \nabla y.B, \Gamma}}{\mathcal{S} \vdash \exists x.A, \nabla y.B, \Gamma}$$

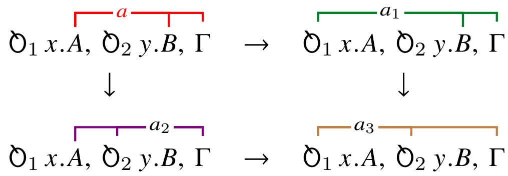
- Case $\exists/\nabla_{\circ}$ with $\nabla \in \{\text{И}, \text{Я}\}$: similar to the previous case, but considering the rule $\nabla_{\circ}$ instead of ∇_{load}.
- Case $\exists/\nabla_{\mathsf{pop}}$ with $\nabla \in \{\text{И}, \text{Я}\}$:



With $\delta_{ab} = \delta_a \setminus \{z\}$, $\delta_{a'} = \delta_a \setminus \{y\}$ and $\delta_{a'b} = (\delta_a \setminus \{z\}) \setminus \{y\}$
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\nabla_{\mathsf{pop}}\dfrac{\exists\dfrac{\mathcal{S} \vdash A[c/z], B, C[x/y], \Gamma}{\mathcal{S} \vdash \exists z.A, B, C[x/y], \Gamma}}{\mathcal{S}, x^{\nabla} \vdash \exists z.A, B, \nabla^{\perp} y.C, \Gamma} \approx \exists\dfrac{\nabla_{\mathsf{pop}}\dfrac{\mathcal{S} \vdash A[c/z], B, C[x/y], \Gamma}{\mathcal{S}, x^{\nabla} \vdash A[c/z], B, \nabla^{\perp} y.C, \Gamma}}{\mathcal{S}, x^{\nabla} \vdash \exists z.A, B, \nabla^{\perp} y.C, \Gamma}$$

- Case $\mathsf{r}_1/\mathsf{r}_2$ with $\mathsf{r}_1, \mathsf{r}_2 \in \{\forall, \nabla_{\mathsf{load}}, \nabla_\circ\}$ with $\nabla \in \{И, Я\}$:



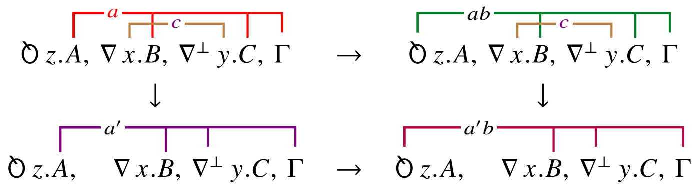
With $\delta_a = \delta_{a_i}$ for $i \in \{1,2,3\}$.
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\mathsf{r}_1\,\dfrac{\mathsf{r}_2\,\dfrac{\mathcal{S} \vdash A, B, \Gamma}{\mathcal{S} \vdash A, \circlearrowleft_2 y.B, \Gamma}}{\mathcal{S} \vdash \circlearrowleft_1 x.A, \circlearrowleft_2 y.B, \Gamma} \approx \mathsf{r}_2\,\dfrac{\mathsf{r}_1\,\dfrac{\mathcal{S} \vdash A, B, \Gamma}{\mathcal{S} \vdash \circlearrowleft_1 x.A, B, \Gamma}}{\mathcal{S} \vdash \circlearrowleft_1 x.A, \circlearrowleft_2 y.B, \Gamma} \quad \text{if } \mathsf{r}_1, \mathsf{r}_2 \in \{\forall, И_\circ, Я_\circ\}$$

$$\mathsf{r}_1\,\dfrac{\mathsf{r}_2\,\dfrac{\mathcal{S}, x^\nabla \vdash A, B, \Gamma}{\mathcal{S}, x^\nabla \vdash A, \circlearrowleft y.B, \Gamma}}{\mathcal{S} \vdash \nabla x.A, \circlearrowleft y.B, \Gamma} \approx \mathsf{r}_2\,\dfrac{\mathsf{r}_1\,\dfrac{\mathcal{S}, x^\nabla \vdash A, B, \Gamma}{\mathcal{S}, x^\nabla \vdash \nabla x.A, B, \Gamma}}{\mathcal{S} \vdash \nabla x.A, \circlearrowleft y.B, \Gamma} \quad \text{if } \mathsf{r}_1 \in \{И_{\mathsf{load}}, Я_{\mathsf{load}}\} \text{ and } \mathsf{r}_2 \in \{\forall, И_\circ, Я_\circ\}$$

$$\mathsf{r}_1\,\dfrac{\mathsf{r}_2\,\dfrac{\mathcal{S}, x^{\nabla_1}, y^{\nabla_2} \vdash A, B, \Gamma}{\mathcal{S}, x^{\nabla_1} \vdash A, \nabla_2 y.B, \Gamma}}{\mathcal{S} \vdash \nabla_1 x.A, \nabla_2 y.B, \Gamma} \approx \mathsf{r}_2\,\dfrac{\mathsf{r}_1\,\dfrac{\mathcal{S}, x^{\nabla_1}, y^{\nabla_2} \vdash A, B, \Gamma}{\mathcal{S}, y^{\nabla_2} \vdash \nabla_1 x.A, B, \Gamma}}{\mathcal{S} \vdash \nabla_1 x.A, \nabla_2 y.B, \Gamma} \quad \text{if } \mathsf{r}_1, \mathsf{r}_2 \in \{И_{\mathsf{load}}, Я_{\mathsf{load}}\}$$

- Case $\nabla_{\mathsf{pop}}/\mathsf{r}$ with $\mathsf{r} \in \{\forall, \nabla_{\mathsf{load}}, \nabla_\circ\}$ and $\nabla \in \{И, Я\}$:

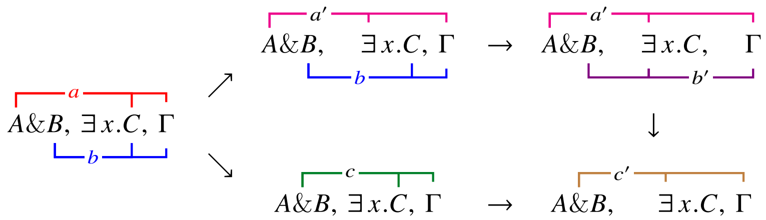


With $\delta_{ab} = \delta_a$, $\delta_{a'} = \delta_a \setminus \{y\}$ and $\delta_{a'b} = \delta_a \setminus \{y\}$
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\nabla_{\mathsf{pop}}\,\dfrac{\mathsf{r}\,\dfrac{\mathcal{S} \vdash A, B, C[x/y], \Gamma}{\mathcal{S} \vdash \circlearrowleft z.A, B, C[x/y], \Gamma}}{\mathcal{S}, x^\nabla \vdash \circlearrowleft z.A, B, \nabla^\perp y.C, \Gamma} \approx \mathsf{r}\,\dfrac{\nabla_{\mathsf{pop}}\,\dfrac{\mathcal{S} \vdash A, B, C[x/y], \Gamma}{\mathcal{S}, x^\nabla \vdash A, B, \nabla^\perp y.C, \Gamma}}{\mathcal{S}, x^\nabla \vdash \circlearrowleft z.A, B, \nabla^\perp y.C, \Gamma} \quad \text{if } \mathsf{r} \in \{\forall, И_\circ, Я_\circ\}$$

$$\nabla^\perp_{\mathsf{pop}}\,\dfrac{\mathsf{r}\,\dfrac{\mathcal{S}, z^{\circlearrowleft} \vdash A, B, C[x/y], \Gamma}{\mathcal{S} \vdash \circlearrowleft z.A, B, C[x/y], \Gamma}}{\mathcal{S}, x^{\nabla^\perp} \vdash \circlearrowleft z.A, B, \nabla y.C, \Gamma} \approx \mathsf{r}\,\dfrac{\nabla^\perp_{\mathsf{pop}}\,\dfrac{\mathcal{S}, z^{\circlearrowleft} \vdash A, B, C[x/y], \Gamma}{\mathcal{S}, z^{\circlearrowleft}, x^{\nabla^\perp} \vdash A, B, \nabla y.C, \Gamma}}{\mathcal{S}, x^{\nabla^\perp} \vdash \circlearrowleft z.A, B, \nabla y.C, \Gamma} \text{if } \mathsf{r} \in \{И_{\mathsf{load}}, Я_{\mathsf{load}}\}$$

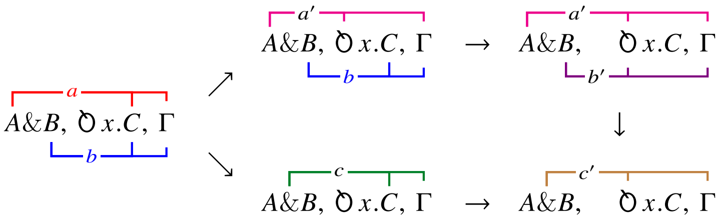
- Case $\&/\exists$:



With $\delta_{a'} = \delta_a \setminus \{y\}$, $\delta_{b'} = \delta_b \setminus \{y\}$, $\delta_c = \delta_a \vee \delta_b$ and $\delta_{c'} = \delta_c \setminus \{y\}$.
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\exists\dfrac{\&\dfrac{\mathcal{S} \vdash A, C[c/x], \Gamma \quad \mathcal{S} \vdash B, C[c/x], \Gamma}{\mathcal{S} \vdash A \mathbin{\&} B, C[c/x], \Gamma}}{\mathcal{S} \vdash A \mathbin{\&} B, \exists x.C, \Gamma} \quad\approx\quad \&\dfrac{\exists\dfrac{\mathcal{S} \vdash A, C[c/x], \Gamma}{\mathcal{S} \vdash A, \exists x.C, \Gamma} \quad \exists\dfrac{\mathcal{S} \vdash B, C[c/x], \Gamma}{\mathcal{S} \vdash B, \exists x.C, \Gamma}}{\mathcal{S} \vdash A \mathbin{\&} B, \exists x.C, \Gamma}$$

- Case $\&/\mathsf{r}$ with $\mathsf{r} \in \{\forall, \nabla_{\mathsf{load}}, \nabla_\circ\}$ with $\nabla \in \{\text{И}, \text{Я}\}$:



With $\delta_{a'} = \delta_a$, $\delta_{b'} = \delta_b$ and $\delta_c = \delta_a \vee \delta_b = \delta_{c'}$.
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\mathsf{r}\dfrac{\&\dfrac{\mathcal{S} \vdash A, C, \Gamma \quad \mathcal{S} \vdash B, C, \Gamma}{\mathcal{S} \vdash A \mathbin{\&} B, C, \Gamma}}{\mathcal{S} \vdash A \mathbin{\&} B, \mho x.C, \Gamma} \quad\approx\quad \&\dfrac{\mathsf{r}\dfrac{\mathcal{S} \vdash A, C, \Gamma}{\mathcal{S} \vdash A, \mho x.C, \Gamma} \quad \mathsf{r}\dfrac{\mathcal{S} \vdash B, C, \Gamma}{\mathcal{S} \vdash B, \mho x.C, \Gamma}}{\mathcal{S} \vdash A \mathbin{\&} B, \mho x.C, \Gamma} \quad \text{if } \mho \in \{\forall, \text{И}_\circ, \text{Я}_\circ\}$$

$$\mathsf{r}\dfrac{\&\dfrac{\mathcal{S}, x^{\mho} \vdash A, C, \Gamma \quad \mathcal{S}, x^{\mho} \vdash B, C, \Gamma}{\mathcal{S}, x^{\mho} \vdash A \mathbin{\&} B, C, \Gamma}}{\mathcal{S} \vdash A \mathbin{\&} B, \mho x.C, \Gamma} \quad\approx\quad \&\dfrac{\mathsf{r}\dfrac{\mathcal{S}, x^{\mho} \vdash A, C, \Gamma}{\mathcal{S} \vdash A, \mho x.C, \Gamma} \quad \mathsf{r}\dfrac{\mathcal{S}, x^{\mho} \vdash B, C, \Gamma}{\mathcal{S} \vdash B, \mho x.C, \Gamma}}{\mathcal{S} \vdash A \mathbin{\&} B, \mho x.C, \Gamma} \quad \text{if } \mathsf{r} \in \{\text{И}_{\mathsf{load}}, \text{Я}_{\mathsf{load}}\}$$

- Case $\mathsf{r}/\#$ with $\mathsf{r} \in \{\exists, \forall, \text{И}_{\mathsf{load}}, \text{Я}_{\mathsf{load}}, \text{И}_\circ, \text{Я}_\circ\}$:

$$\begin{array}{ccc}
\#(a_1, \dots, a_l, b_1, \dots, b_k, c) & \rightarrow & \#(\#(a_1, \dots, a_l), \#(b_1, \dots, b_k, c)) \\
\downarrow & & \downarrow \\
\#(a_1, \dots, a_l, b_1, \dots, b_k, c') & \rightarrow & \#(\#(a_1, \dots, a_l), \#(b_1, \dots, b_k, c'))
\end{array}$$

Where either
- $\mathsf{r} \in \{\forall, \text{И}_\circ, \text{Я}_\circ, \text{И}_{\mathsf{load}}, \text{Я}_{\mathsf{load}}\}$, thus $c = \Gamma, A$, $c' = \Gamma, \mho x.A$, and $\delta_{c'} = \delta_c$; or
- $\mathsf{r} \in \{\exists, \text{И}_{\mathsf{pop}}, \text{Я}_{\mathsf{pop}}\}$, thus $c = \Gamma, A$, $c' = \Gamma, \exists x.A$ and $\delta_{c'} = \delta_c \setminus \{x\}$.

The derivation labelling the link in the bottom-right corner of the diagram is the same, independently of the sequence of coalescence steps.

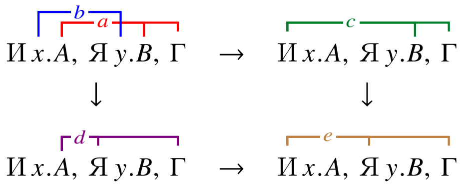
- Case mix/mix :

  if this case applies, then the co-tree has a $\frown$-node x with leaves on which only (mix) can be applied to merge some of them. In particular, x must have at least three leaves a, b and c such that, without loss of generaltity, a (mix) can be appliet to merge a and b, or to merge b and c. Let ab be the link obtained by applying a (mix) step to merge a and b. To conclude it suffices to remark that we can always find a continuation of the coalescence path containing such step (mix) which also contain another step (mix) merging c and the link obtained by applying coalescence steps involving ab. This follows from the fact that the side condition of the step (mix) implies that, if exists, the least common ancestor (in the forest of Γ) of two formulas from two links which could be have been merged using a step (mix) must be a $\parr$. Thus, under the conditio that $\tau(\Lambda)$ is a proof net, once we cannot apply any more coalescence step to a link obtained from ab, we are still able to apply a (mix) step to merge it with c.

- Case $\blacktriangleleft_\circ$ /$\blacktriangleleft_\circ$: similar to the previous case.

Finally, we have the following special additional cases, in which the dualizer is not the same, but the proof nets are.

- Case $\text{И}_{\mathsf{pop}}/\text{Я}_{\mathsf{pop}}$:



  where $\delta_c = \delta_d = \delta_e = \delta_a \setminus \{y\}$.
  The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\mathcal{D}_1 \;=\; \text{И}_{\mathsf{load}}\dfrac{\text{И}_{\mathsf{pop}}\dfrac{\mathcal{S} \vdash \Gamma, A, B[x/y]}{\mathcal{S}, x^{\text{И}} \vdash \Gamma, A, \text{Я}y.B}}{\mathcal{S} \vdash \Gamma, \text{И}x.A, \text{Я}y.B} \;\approx_{\mathsf{w}}\; \text{Я}_{\mathsf{load}}\dfrac{\text{Я}_{\mathsf{pop}}\dfrac{\mathcal{S} \vdash \Gamma, A[y/x], B}{\mathcal{S}, y^{\text{Я}} \vdash \Gamma, \text{И}x.A, B}}{\mathcal{S} \vdash \Gamma, \text{И}x.A, \text{Я}y.B} \;=\; \mathcal{D}_2$$

  Where $\mathcal{D}_1 \approx_{\mathsf{w}} \mathcal{D}_2$ because

$$\mathcal{D}_1 \;\sim_{\mathsf{w}}\; \text{И}_{\mathsf{load}}\dfrac{\text{И}_{\mathsf{pop}}\dfrac{\mathcal{S} \vdash \Gamma, A[z/x], B[z/y]}{\mathcal{S}, z^{\text{И}} \vdash \Gamma, A[z/x], \text{Я}y.B}}{\mathcal{S} \vdash \Gamma, \text{И}x.A, \text{Я}y.B} \;\approx\; \text{Я}_{\mathsf{load}}\dfrac{\text{Я}_{\mathsf{pop}}\dfrac{\mathcal{S} \vdash \Gamma, A[z/x], B[z/x]}{\mathcal{S}, z^{\text{Я}} \vdash \Gamma, \text{И}x.A, B[z/x]}}{\mathcal{S} \vdash \Gamma, \text{И}x.A, \text{Я}x.B} \;\sim_{\mathsf{w}}\; \mathcal{D}_2$$

◀

▶ Remark 53. As already observed in [52], coalescence is not confluent on non-coalescent co-tree, as shown in the following examples.