Proof Nets for the π-Calculus

Matteo Acclavio
University of Sussex

Giulia Manara
University of Southern Denmark

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.

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, Deduction-as-computation, Proof equivalence, Proof nets, π-calculus

Digital Object Identifier 10.4230/LIPIcs...

Funding *Matteo Acclavio*: This work was partially supported by the Villum Fonden grant no. 50079

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 tracesthat differ due to factors beyond the control of the program, such as race conditions and side effects.

In this work, we propose a logical framework based on the *deduction-as-computation* interpretation of proofs for the π-calculus, providing a formalism satisfying these three desiderata.

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 [60] 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 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 [47, 46, 48, 77] 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 [38] for MLL with units, [50, 48] 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.

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

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 [81], we would say they are in conflict relation.

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.

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

[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.

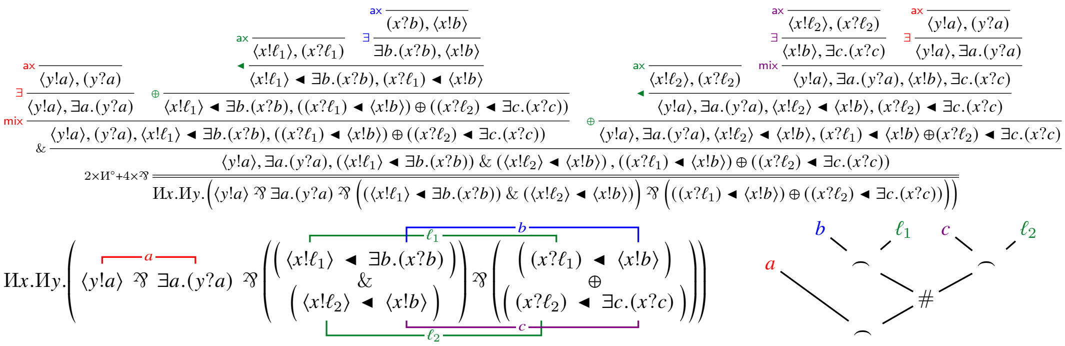


Figure 1 A derivation of the formula $[\![P]\!]$ corresponding to computation tree in the right of Figure 4, and the corresponding proof net.

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 computation trees of a process can be interpreted as derivations of the corresponding formula. We also define proof nets for PiL by combining the techniques used in *conflict nets* [38] and in *unification nets* [49, 39]. As a byproduct, we also define the first syntax of conflict nets for MALL^1. By combining the correspondence between computation trees and derivations, and between derivations modulo local rule permutations and proof nets, we provide canonical representatives of computation trees modulo interleaving.

Related Works. Following the ideas in [60], Miller proposed in [59] 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 $⅋$ and the choice operator $+$ in the original formulation of the π-calculus [64] by the $\oplus$. Guglielmi developed an extension of multiplicative linear logic with a non-commutative connective aiming at internalizing sequentiality in [35, 36], lately leading to the design *deep inference* and the formalism of the *calculus of structures* [37] to obtain a satisfactory proof system for the logic BV. In [21] Bruscoli established a *computation-as-deduction* for a simple fragment of CCS (without recursion and choice) where successful terminating executions of a process correspond to specific derivations in BV. This correspondence has been extended to the π-calculus by Horne, Tiu et al [43, 42, 45], including the choice operator ($+$), modeled via the additive connective $\oplus$, and name restriction, modeled using *nominal quantifiers* in the spirit of [68, 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 [78];
- as Horne and Tiu in [43, 42, 45], we use a pair of dual nominal quantifiers (instead of a self-dual quantifier as in [58, 73, 62]) to model restriction. This because the use of a self-dual nominal quantifier would allow to encode processes with complete different computational property like $(\nu x)\,(x!\langle a\rangle.\mathsf{Nil} \mid y?(a).\mathsf{Nil})$ (which is deadlock-free) and $(\nu x)\,(y?(a).\mathsf{Nil}) \mid (\nu x)\,(x!\langle a\rangle.\mathsf{Nil})$ (which is stuck) with logically equivalent formulas. However, as explained in detail in Remark 9 and in Appendix A, our pair of dual quantifiers

satisfies different proof theoretical properties;

- in [44, 45] Horne et al use the original version of the π-calculus [64] which feature a choice operator + with an undesirable "non-local" behavior, which requires to forwardly check that it will entail a communication rule[2]. Its rule is written as follows

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

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. For example, even if one would expect that the process $P + \mathsf{Nil}$ could choose to continue as Nil, this is not allowed by the semantics since the choice is constrained by the possibility of performing an action after a choice, and Nil can perform no actions.

A logical connective 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 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 [80, 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 [80, 32, 41] and choreographic programming [65].[3]

- in the work of Bruscoli [21], and in the works of Horne and Tiu [45, 42] derivations correspond to executions, while in our work derivations represent computation trees. However, if we restrict the label-send constructor $x \triangleleft \{\ell : P_\ell\}_{\ell \in L}$ on singleton sets of labels (or equivalently pruning all branching except one of each &-rule), we can recover such a correspondence between derivations and executions.

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 computation trees of a process P can be represented by derivations of the corresponding formula. Thereby, we show that equivalent computation 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] This is not a problem in their setting because they are interested in establishing a correspondence between terminating executions only (i.e., reducing to Nil) and proofs.

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

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 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., [79]), 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 $\mathsf{r}\,\dfrac{}{\mathcal{S} \vdash \Gamma}$, $\mathsf{r}\,\dfrac{\mathcal{S} \vdash \Gamma_1}{\mathcal{S} \vdash \Gamma}$, or $\mathsf{r}\,\dfrac{\mathcal{S} \vdash \Gamma_1 \quad \mathcal{S} \vdash \Gamma_2}{\mathcal{S} \vdash \Gamma}$. 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 the conclusion of a rule in X, whose children are (all and only) the premises of the rule. An ***open premises*** 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 $\begin{array}{c}\mathcal{D}\,\Vert \\ \mathcal{S} \vdash \Gamma\end{array}$ $\left(\text{resp.}\ \begin{array}{c}\mathcal{S} \vdash \Delta \\ \mathcal{D}\,\Vert \\ \mathcal{S} \vdash \Gamma\end{array}\right)$.

2.1 π-Calculus

We consider the version of π-calculus presented in [80, 32], whose processes are generated from a countable set of ***(channel) names*** $\mathcal{N} = \{x, y, \dots\}$ and (disjoint) finite set of ***labels*** $\mathcal{L}$ grammar in Figure 2. 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

[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.

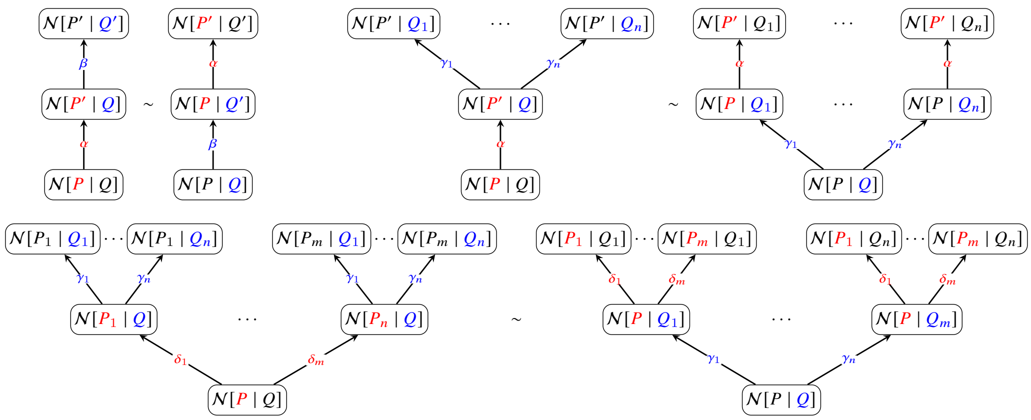


Figure 3 Generators of the computation tree equivalence with $\alpha, \beta \in \{\mathsf{Com}, \mathsf{Sel}\}$ and $\gamma_i, \delta_j \in \{\mathsf{Bra}\}$, where $\{P_1, \ldots, P_m\}$ (resp. $\{Q_1, \ldots, Q_n\}$) is the set of all processes such that $P \to P_i$ (resp. $Q \to Q_j$) via Bra.

$x_1, \ldots x_n$ 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:

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

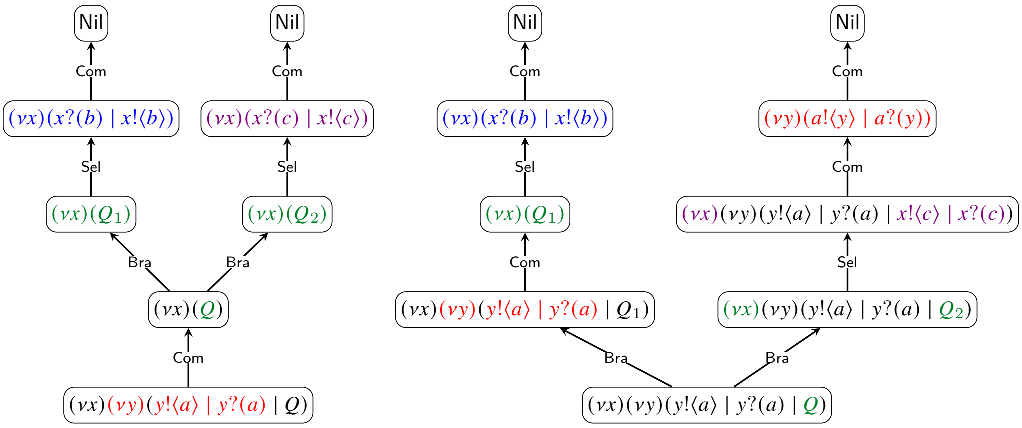
A ***computation trees*** of a process P is a trees of processes with root P, where a process Q' is a child of Q if $Q \to Q'$, and such that branching is determined by the intrinsic non-determinism of the reduction rule Bra, that is, if two processes Q_1 and Q_2 are children of a same process Q, then $Q \to Q_1$ 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 computation tree with the unique reduction rule in $\{\mathsf{Com}, \mathsf{Bra}, \mathsf{Sel}\}$ required to reduce the term P to Q.[5] The ***interleaving*** equivalence relation ($\sim$) on computation trees is defined by the relations in Figure 3. See Figure 4 for an example of two computation trees equivalent modulo interleaving.

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 [64] (but also in CCS [63]). Even if it would be desirable to require $\blacktriangleleft$ to be

[5] 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 computation tree, see the definition of *core-reduction* in [8].



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

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

Formulas		De Morgan Laws	α-equivalence
$A, B \coloneqq \circ$	unit (atom)		
$\mid \langle x!y\rangle$	atom		
$\mid (x?y)$	atom	$\circ^\perp = \circ$	$a = a$
$\mid A \parr B$	par	$(A^\perp)^\perp = A$	if $a \in \{\circ, \langle x!y\rangle, (x?y)\}$
$\mid A \otimes B$	tensor	$\langle x!y\rangle^\perp = (x?y)$	
$\mid A \blacktriangleleft B$	prec	$(A \parr B)^\perp = A^\perp \otimes B^\perp$	$A_1 \odot A_2 = B_1 \odot B_2$
$\mid A \oplus B$	oplus	$(A \blacktriangleleft B)^\perp = A^\perp \blacktriangleleft B^\perp$	if $A_i = B_i$
$\mid A \& B$	with	$(A \oplus B)^\perp = A^\perp \& B^\perp$	and $\odot \in \{\parr, \blacktriangleleft, \otimes, \oplus, \&\}$
$\mid \forall x.A$	for all	$(\forall x.A)^\perp = \exists x.A^\perp$	
$\mid \exists x.A$	exists	$(\text{И}X.A)^\perp = \text{Я}X.A^\perp$	$\text{Ꝺ}x.A = \text{Ꝺ}y.A[y/x]$
$\mid \text{И}x.A$	new		y fresh for $A$and $\text{Ꝺ} \in \{\text{И}, \text{Я}, \forall, \exists\}$
$\mid \text{Я}x.A$	ya		

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

associative, to capture the associativity of sequential composition of processes, we instead let $\blacktriangleleft$ 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, but allowing us to use the existential (and universal) quantifiers in the standard way.

▶ Remark 1. In our processes-as-formulas translation (see Section 7), we use the existential quantifiers to bind variables used as input of a communication to capture name passing, and the nominal quantifier И to model restriction. As already explained in [62], 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 quanfitication, then any property for Q should also be valid for R, because $\forall x.\forall y.P(x, y)$ entails $\forall z.P(z, z)$.

Moreover, if universal quantification would be used to model restriction, it would clash with the use of the existential quantifier to model name passing because of the duality between these two quantifiers.

▶ Remark 2. In our work, we do not consider a self-dual nominal quantifier as the one studied in [62, 23, 73], but we rather introduce a dual quantifier similarly to what is done in [43, 42, 45], where such a design choice is justified in view of the semantics of the π-calculus.

In [45] the authors list the three following logical properties a nominal quantifier Ꝺ modeling the binder should satisfy:

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

However, we claim that the following additional condition (which holds in their systems too) should be added in this list, in view of how restriction and choice operators interact.

4 *name-choice*: if $\odot$ is a connective modeling a global choice, then $\text{Ꝺ}x.A \odot \text{Ꝺ}x.B$ should imply $\text{Ꝺ}x.(A \odot B)$.

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. In our work, all these requirements are met (see Proposition 12).

▶ **Definition 3.** ***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. As standard, we assume formulas to be written using* ***Barendregt's convention****, that is, each variable x occurs bounded by at most a quantifier and, if bounded, cannot occur free in the same formula. 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 $\mathsf{free}(\Gamma) = \bigcap_{i=1}^n \mathsf{free}(A_i)$.*

▶ Remark 4. 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 5.** *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 $\mathcal{S}$.*

▶ **Notation 6.** *We write judgements $\mathcal{S} \vdash \Gamma$ with $\mathcal{S} = \emptyset$ (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 denote by $\mathcal{S}_1, \mathcal{S}_2$ the union of two disjoint stores $\mathcal{S}_1$ and $\mathcal{S}_2$.*

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

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

$$\oplus\frac{\mathcal{S} \vdash \Gamma, A_i}{\mathcal{S} \vdash \Gamma, A_1 \oplus B_2} \quad \&\frac{\mathcal{S} \vdash \Gamma, A \quad \mathcal{S} \vdash \Gamma, B}{\mathcal{S} \vdash \Gamma, A \& B} \quad \forall\frac{\mathcal{S} \vdash \Gamma, A}{\mathcal{S} \vdash \Gamma, \forall x.A}\dagger \quad \exists\frac{\mathcal{S} \vdash \Gamma, A[y/x]}{\mathcal{S} \vdash \Gamma, \exists x.A}$$

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

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

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

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

Figure 6 Sequent calculus rules with side conditions $\dagger \coloneqq x \notin \mathsf{free}(\Gamma)$. The rules above the double line form the system PiL.

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

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

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

▶ Remark 8. During proof search, the rule ∇_{load} allow to remove an occurrence of the nominal quantifier ∇ under the condition of storing the variable bound by the nominal quantifier in the store. Since the axiom rule and the unit rule have empty store, if the proof search is successful, then such a variable has to be used for the substitution of a variable bound by the dual nominal quantifier $\nabla^{\perp}$. That is, each ∇_{load} rule is paired with some ∇_{pop} rules above it in a derivation. Note that in absence of additive connectives, such a ∇_{pop} is unique. This pairing can be seen as a long-distance link between the nominal quantifiers ∇ and $\nabla^{\perp}$ in a provable sequent. This same link is more clear if looking at the rule И-Я introducing a pair of dual nominal quantifiers binding a same variable x. However, this latter rule is weaker, and, in particular, does not suffice to prove quantifier swap equivalences for nominal quantifiers and the nominal-choice laws from Equation (5).

▶ Remark 9. 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 [43, 42, 45]. One difference is the way И and Я interact in PiL, in which each nominal quantifier ♉ interacts with at most one dual quantifier $♉^{\perp}$, 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 exactly 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 [51, 54, 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, in PiL the implication $Иx.A \multimap (Иx.A^{\perp})^{\perp}$ (that is, the formula $Яx.A \parr Яx.A$), while in BV^1 the same formula is derivable.

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 $Иx.(A \blacktriangleleft B) \multimapboth (Иx.A) \blacktriangleleft B$ 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, 61], 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 10.** *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 $Я_{\mathsf{pop}}$. ◀

▶ **Proposition 11.** *Then the rule* AX *is derivable in* mini-PiL*. Therefore, also in* PiL^- *and* PiL.

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

- 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 $A = B \blacktriangleleft C$, then $A^\perp = B^\perp \blacktriangleleft C^\perp$. We apply a rule $\blacktriangleleft$, and we conclude by inductive hypothesis;
- if $A = \mho x.B$ with $\nabla \in \{\text{И}, \text{Я}\}$, 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 complete statement follow from Proposition 10. ◀

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 12.** *The following logical equivalences and implications are derivable in* PiL.

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

(5)

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

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

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

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

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

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.

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

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

◄

► Remark 13. The system mini-PiL satisfies all the equivalences and implications in Proposition 12, except for quantifier swap and nominal choices, while the system PiL^- satisfies all the equivalences and implications in Proposition 12, except for the bottom-most nominal choice. That is, in $\vdash_{\mathsf{PiL}^-} (\text{И}x.A \oplus \text{И}x.B) \multimap (\text{И}x.(A \oplus B))$ but not the converse implication.

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

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

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

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

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.

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

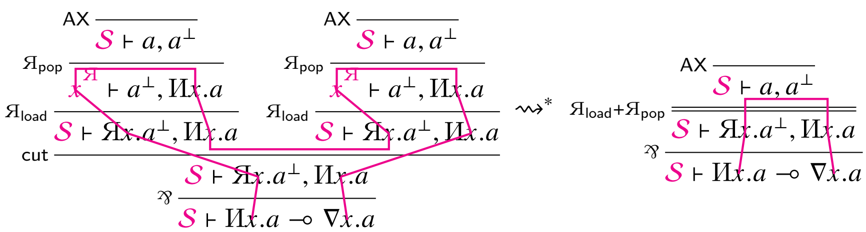
$$
{\scriptstyle\mathsf{cut}}\dfrac{{\scriptstyle\blacktriangleleft}\dfrac{\mathcal{S}_1\vdash\Gamma_1,C,A \quad \mathcal{S}_2\vdash\Gamma_2,D,B}{\mathcal{S}_1,\mathcal{S}_2\vdash\Gamma_1,\Gamma_2,C\blacktriangleleft D,A\blacktriangleleft B} \quad {\scriptstyle\blacktriangleleft_{\circ}}\dfrac{\mathcal{S}_3\vdash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

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

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

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 8. For example, consider the derivation with cut in the left of Equation (9) of $\mathcal{S}\vdash \text{И}x.a \multimap \nabla x.a$, where we marked the flows of the nominal variables.



(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 ***name linking*** rule we use during the rewriting process of cut-elimination.

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

▶ **Theorem 14** (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 13 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 15. 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 [70] and BV [37], 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 **??**.

▶ Remark 16. 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 $\mathsf{PiL} \setminus \{\text{И}_{\mathsf{pop}}, \text{Я}_{\mathsf{load}}\}$ presented in [55] also satisfy cut-elimination, and it already expressive enought to support the interpreting derivations as computation trees (see Section 7). Indeed, the only difference is that in such a system the nominal-choice law $(\text{И}x.A \oplus \text{И}x.B) \multimap\!\circ (\text{И}x.(A \oplus B))$ does not hold, but onlu the left-to-right implication is provable (see the right branch of the bottom-most derivation in Equation (8)).

▶ **Corollary 17.** *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 $\vdash_{\mathsf{PiL}} A \multimap C$.*

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

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

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

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

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

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

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 $\nabla_{\circ}$.

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 ⅋ 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 ⅋-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. ◀

We conclude this section stating that PiL can be embedded in MAV^1 [45] using a translation $\lceil\cdot\rceil$ replacing each occurrence of ◀ with a ◁, and each occurrence of Я with a ∍. Formal definitions and details of the proof are provided in Appendix A.

▶ **Theorem 18.** *Let $A_1, \ldots, A_n$ be formulas. If $\vdash_{\mathsf{PiL}} A_1, \ldots, A_n$, then $\vdash_{\mathsf{MAV}^1} \parr_{i=1}^{n} \lceil A_i \rceil$.*

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 ◄) and additive ($\oplus$ and $\&$) connectives in PiL in a canonical way, we follow the approach used in Heijtjes and Hughes' *conflict nets* [48], providing canonical representative for proofs modulo local rule permutations rather than Hughes and van Glabbeek's *slice nets* [50] or Girard's *monomial nets* [34, 53]. This is because of the well-known trade-off in terms of complexity between proof translation, sequentialization and cut-elimination (see page 3 of [48]). Conflict nets are optimal for sequentialization and proof translation, but not with respect to cut-elimination – and in our paradigm the rule cut, and cut-elimination, plays no role. Thereby, we opt for an optimal solution for the aspects that are relevant to the application we aim at in this paper (see Section 6). Intuitively, conflict nets for MALL is a tree alternating *concord* ($\frown$) and *conflict* ($\#$) nodes, with leaves *axiom links*, and satisfying a contractility criterion with respect to a rewriting called *coalescence*.

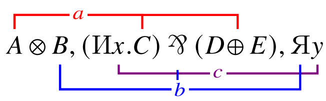
To handle quantifiers, we use the approach adopted in *unification net* for MLL [49] by Hughes, and additive linear logic [39] by Heijtjes, Hughes and Straßburger. This allows us to provide proof nets satisfying the principle of *generality* (in the sense of Lambek [52]), that is, identifying proofs differing in the witness assigned to quantifiers. Intuitively, unification nets are defined as sets of axiom links labeled by a substitution (called *dualizer*) that instantiates each variable with the name used in the proof.

▶ **Notation 19.** *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*[6] *with* ***domain*** *$\{y_1, \ldots, y_n\}$. Moreover, we use the following denotations:*

- *$\sigma\tau$ is the* ***composition*** *of σ and τ (in which σ is applied after τ);*
- *$\sigma \setminus \{x\}$ 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 $\tau \leq \rho$;*

A ***pre-link*** a on a sequent Γ is either a ***nominal link***, that is, a pair $\{x, y\}$ of variables occurring in Γ with x bounded by И and y by a Я, or a sequent which is an induced sub-forest of Γ. We represent a pre-link a by drawing a horizontal line connected via vertical segments (labeled by a) to the roots of each variable or subformula in the link.

▶ **Example 20.** Consider the sequent $\Gamma = A \otimes B, (\text{И}x.C) \parr (D \oplus E), \text{Я}y.F$. The sequent $\Gamma' = D, D \oplus E$ is not a pre-link because the formula D is repeated twice. We represent the pre-links $a = A, C, D \oplus E$ and $b = B, \forall x.F, (\text{И}x.C) \parr (D \oplus E)$ and $c = \{x, y\}$ as shown below.



[6] 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 [39].

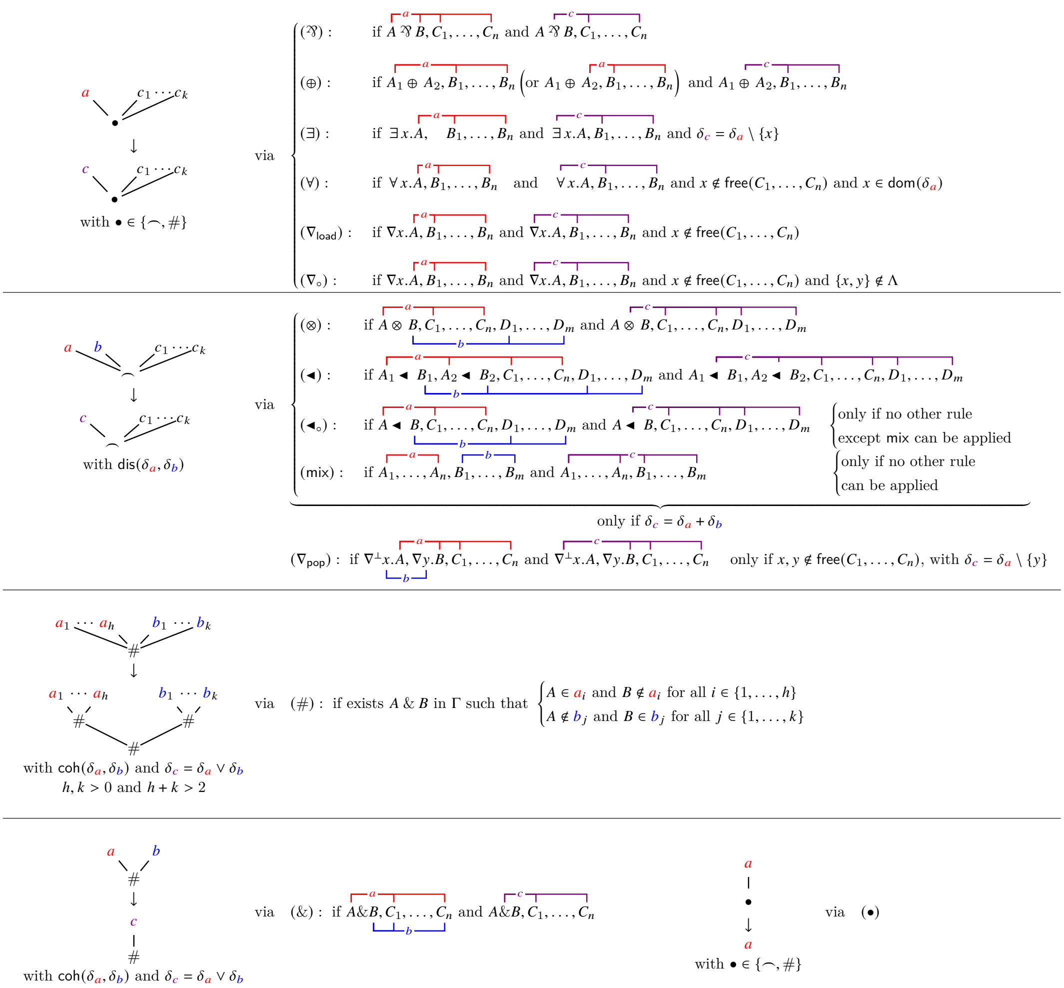


Figure 10 Coalescence steps for co-trees, with $\nabla \in \{\text{И}, \text{Я}\}$ and $a, b, c_1, \ldots, c_n$ leaves.

▶ **Definition 21.** *Let Γ be a sequent. A **dualizer** σ for Γ is a substitution with domain variables occurring in Γ bounded by an existential quantifier ($\exists$) or a nominal quantifier Я. A **link** on Γ is a pre-link a equipped with a (possibly empty) dualizer δ_a for Γ.*

*A **concord-conflict tree** (or **co-tree** for short) on Γ is a tree Λ with leaves labeled by links, and internal nodes labeled by $\frown$ (**concord** nodes) or by $\#$ (**conflict** nodes). It is **axiomatic** if it contains only **axiomatic links**, that is, links made of a single occurrence of $\circ$, a pair of variables $\{x, y\}$, or a pair of the atoms of the form $\{\langle x!y\rangle, (z?t)\}$. We denote by $\lfloor\Lambda\rfloor$ the co-tree obtained by merging adjacent $\frown$-nodes (resp. $\#$-nodes) nodes, and by removing node with a single child (by attaching its child to its parent). A co-tree Λ is **canonical** if $\Lambda = \lfloor\Lambda\rfloor$. We may denote $\Lambda_1 \frown \Lambda_2$ or $\frown(\Lambda_1, \ldots, \Lambda_n)$ (resp. $\Lambda_1 \# \Lambda_2$ or $\#(\Lambda_1, \ldots, \Lambda_n)$) a co-tree with root a $\frown$-node (resp. $\#$-node) and with children roots of $\Lambda_1, \ldots, \Lambda_n$.*

▶ **Definition 22.** *We define **coalesce steps** over co-trees in Figure 10. A co-tree Λ **coalesces** to Λ' if $\Lambda \to \Lambda'$, and it is **coalescent** there is a **coalescence path** (i.e., a sequence of coalescent steps) such that Λ rewrites to a co-tree made of a single-link with empty dualizer.*

*A **proof net** for a sequent Γ is a coalescent axiomatic co-tree Λ on Γ. We say that two proof nets Λ_1 and Λ_2 are **isomorphic** (denoted $\Lambda_1 = \Lambda_2$) if they are the the same co-tree.*

To prove the soundness and completeness of proof nets for PiL, we define a *desequential-*

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

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	$\mathcal{D}_c$
$\begin{array}{c}\pi_1 \Vert \\ \delta_a(\mathcal{S}_1 \vdash A, \Gamma)\end{array}$	$\begin{array}{c}\pi_2 \Vert \\ \delta_b(\mathcal{S}_2 \vdash B, \Delta)\end{array}$	$\xrightarrow{\otimes}$	$\otimes\ \dfrac{\begin{array}{cc}\pi_1 \Vert & \pi_2 \Vert \\ \delta_a(\mathcal{S}_1 \vdash A, \Gamma) & \delta_b(\mathcal{S}_2 \vdash B, \Delta)\end{array}}{\delta_c(\mathcal{S}_1, \mathcal{S}_2 \vdash A \otimes B, \Gamma, \Delta)}$
$\begin{array}{c}\pi_1 \Vert \\ \delta_a(\mathcal{S}_1 \vdash A, \Gamma)\end{array}$	$\begin{array}{c}\pi_2 \Vert \\ \delta_b(\mathcal{S}_2 \vdash B, \Delta)\end{array}$	$\xrightarrow{\blacktriangleleft_\circ}$	$\blacktriangleleft_\circ\ \dfrac{\begin{array}{cc}\pi_1 \Vert & \pi_2 \Vert \\ \delta_a(\mathcal{S}_1 \vdash A, \Gamma) & \delta_b(\mathcal{S}_2 \vdash B, \Delta)\end{array}}{\delta_c(\mathcal{S}_1, \mathcal{S}_2 \vdash A \blacktriangleleft B, \Gamma, \Delta)}$
$\begin{array}{c}\pi_1 \Vert \\ \delta_a(\mathcal{S}_1 \vdash A_1, A_2\Gamma)\end{array}$	$\begin{array}{c}\pi_2 \Vert \\ \delta_b(\mathcal{S}_2 \vdash B_1, B_2\Delta)\end{array}$	$\xrightarrow{\blacktriangleleft}$	$\blacktriangleleft\ \dfrac{\begin{array}{cc}\pi_1 \Vert & \pi_2 \Vert \\ \delta_a(\mathcal{S}_1 \vdash A_1, A_2, \Gamma) & \delta_b(\mathcal{S}_2 \vdash B_1, B_2, \Delta)\end{array}}{\delta_c(\mathcal{S}_1, \mathcal{S}_2 \vdash A_1 \blacktriangleleft B_1, A_2 \blacktriangleleft B_2, \Gamma, \Delta)}$
$\begin{array}{c}\pi_1 \Vert \\ \delta_a(\mathcal{S}_1 \vdash \Gamma)\end{array}$	$\begin{array}{c}\pi_2 \Vert \\ \delta_b(\mathcal{S}_2 \vdash \Delta)\end{array}$	$\xrightarrow{\mathsf{mix}}$	$\mathsf{mix}\ \dfrac{\begin{array}{cc}\pi_1 \Vert & \pi_2 \Vert \\ \delta_a(\mathcal{S}_1 \vdash \Gamma) & \delta_b(\mathcal{S}_2 \vdash \Delta)\end{array}}{\delta_c(\mathcal{S}_1, \mathcal{S}_2 \vdash \Gamma, \Delta)}$
$\begin{array}{c}\pi_1 \Vert \\ \delta_a(\mathcal{S} \vdash A, \Gamma)\end{array}$	$\begin{array}{c}\pi_2 \Vert \\ \delta_b(\mathcal{S} \vdash B, \Delta)\end{array}$	$\xrightarrow{\&}$	$\&\ \dfrac{\begin{array}{cc}\pi_1 \Vert & \pi_2 \Vert \\ \delta_a(\mathcal{S} \vdash A, \Gamma) & \delta_b(\mathcal{S} \vdash B, \Delta)\end{array}}{\delta_c(\mathcal{S} \vdash A \,\&\, B, \Gamma, \Delta)}$

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

Figure 11 Effect of coalescence steps in Figure 10 on co-trees with leaves labeled by derivations. The steps $\bullet$ and $\#$ change no link labels.

ization procedure, mapping derivations to proof nets, and a *sequentialization*, mapping proof nets to co-trees.

▶ **Theorem 23.** *Let Γ be a sequent. Then $\vdash_{\mathsf{PiL}} \Gamma$ iff there is a* PiL*-net Λ on Γ.*

Proof. Let $\mathcal{D}$ be a derivation of Γ in PiL. We defined the (axiomatic) co-tree $[\![\mathcal{D}]\!]$ by translating top-down rules $\mathcal{D}$ as shown in Figure 12, and we let $\Lambda_{\mathcal{D}}$ be the canonical co-tree associated to $[\![\mathcal{D}]\!]$, that is, the co-tree obtained by removing any single-child internal node and by merging two adjacent internal nodes with the same label in $[\![\mathcal{D}]\!]$. Prove that $\Lambda_{\mathcal{D}}$ 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, $\blacktriangleleft_\circ$ and И-Я may required 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 whose leaves (which are links) are labeled by derivations with conclusion the link itself. Given a proof net Λ, we associate to it a deductive co-tree whose leaves are labeled by ax- or $\circ$-rules. Since Λ is coalescent, we can define a derivation $\mathcal{D}_\Lambda$ in PiL with conclusion Γ using a given coalescence path by modifying the derivations labeling the leaves of Λ as shown in Figure 11. ◀

As in [48], we define the ***size*** of a proof net Λ on Γ as the number $|\Lambda|$ of nodes in the co-tree Λ plus the number $|\Gamma|$ of nodes in the forest Γ.

▶ **Proposition 24.** *The coalescence criterion is polynomial in the size of the proof net.*

Proof. The result follows from the same argument (and algorithms) used in [48] in the proof of the similar result for MALL. The new multiplicative coalescence steps (the ones involving the $\blacktriangleleft$) are as complex as the $\otimes$. Coalescence steps involving quantifiers requires to perform operations on dualizers which are linear in the size of the dualizer (see [56]), 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. ◀

$$\left\{\!\!\left\{\circ\,\frac{}{\mathcal{S} \vdash \circ}\right\}\!\!\right\} = \left\{\overset{a}{\circ}\right\} \qquad \left\{\!\!\left\{\mathsf{ax}\,\frac{}{\mathcal{S} \vdash \langle x!y\rangle, (x?y)}\right\}\!\!\right\} = \left\{\overbrace{\langle x!y\rangle\;,\;(x?y)}^{a}\right\} \qquad \left\{\!\!\left\{\nabla_{\mathsf{pop}}\,\frac{\begin{array}{c}\mathcal{D}_1 \\ \mathcal{S} \vdash \Gamma, A[y/x]\end{array}}{\Sigma, y^{\nabla} \vdash \Gamma, \nabla^{\perp}x.A}\right\}\!\!\right\} = \left\lfloor \{\!\{\mathcal{D}_1\}\!\}\,[y/x] \frown \left\{\overbrace{x\;,\;y}^{a}\right\}\right\rfloor$$

$$\left\{\!\!\left\{\mathsf{r}^1\,\frac{\begin{array}{c}\mathcal{D}_1 \\ \mathcal{S} \vdash \Gamma_1\end{array}}{\mathcal{S} \vdash \Gamma}\right\}\!\!\right\} = \{\!\{\mathcal{D}_1\}\!\} \qquad \left\{\!\!\left\{\nabla_{\mathsf{load}}\,\frac{\begin{array}{c}\mathcal{D}_1 \\ \mathcal{S}, x^{\nabla} \vdash \Gamma, A\end{array}}{\mathcal{S} \vdash \Gamma, \nabla x.A}\right\}\!\!\right\} = \{\!\{\mathcal{D}_1\}\!\} \qquad \left\{\!\!\left\{\exists\,\frac{\begin{array}{c}\mathcal{D}_1 \\ \mathcal{S} \vdash \Gamma, A[y/x]\end{array}}{\mathcal{S} \vdash \Gamma, \exists x.A}\right\}\!\!\right\} = \{\!\{\mathcal{D}_1\}\!\}\,[y/x]$$

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

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

Figure 12 Translation of a derivation in PiL into a proof net, where $\{\!\{\mathcal{D}\}\!\}\,[y/x]$ is the co-tree obtained by applying the substitution $[y/x]$ to all its links and by letting the dualizers δ_a in $\{\!\{\mathcal{D}\}\!\}$ being $\delta_a[y/x]$ in $\{\!\{\mathcal{D}\}\!\}\,[y/x]$.

6 Canonicity results

In this section we prove that proof nets for PiL provide canonical representative for derivations modulo local rule permutations, and that two derivations obtained by sequentializing the same proof net are equivalent modulo local rule permutations and what we refer to as *witness renaming*.

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

▶ **Definition 25.** *The **variable replacement** of an instance of a quantifier rule in a derivation $\mathcal{D}$ is the variable*[7] *used to replace the variable bound by the quantifier in the conclusion of the rule. Using the convention for rules from Figure 6, the variable replacement of a rule $\exists$, И_{pop}, or Я_{pop} is the witness y used int the substitution in the premise of the rule, while the variable replacement of a rule $\forall$, $\text{И}_{\circ}$, $\text{Я}_{\circ}$, И_{load}, or Я_{load} is variable bound by the quantifier removed by such a rule.*

Two derivations $\mathcal{D}_1$ and $\mathcal{D}_2$ in PiL are:

- ***equivalent modulo fresh names** (denoted $\mathcal{D}_1 \sim_{\mathsf{w}} \mathcal{D}_2$) if it is possible to tranform $\mathcal{D}_1$ into $\mathcal{D}_2$ by changing the variable replacements of quantifier rules, and propagating the changes upwars in the derivation;*
- ***equivalent modulo local rule permutations** (denoted $\mathcal{D}_1 \sim \mathcal{D}_2$) if it is possible to tranform $\mathcal{D}_1$ into $\mathcal{D}_2$ using the transformations in Figure 13;*
- ***equivalent modulo local rule permutations and withness choices** (denoted $\mathcal{D}_1 \approx \mathcal{D}_2$) if there are derivations $\mathcal{D}_1'$ and $\mathcal{D}_2''$ such that $\mathcal{D}_1 \sim_{\mathsf{w}} \mathcal{D}_1' \sim \mathcal{D}_2' \sim_{\mathsf{w}} \mathcal{D}_2$.*

▶ Remark 26. In Equation (11), the derivation on the right is obtained by renaming the principal name x of the quantifier with a z using α-equivalence, and then propagating the change upwards.

$$\text{И}_{\mathsf{load}}\,\dfrac{\text{И}_{\mathsf{pop}}\,\dfrac{\mathsf{ax}\,\dfrac{}{\vdash \langle x!a\rangle, (x?a)}}{x^{\text{И}} \vdash \langle x!a\rangle, \text{Я}y.(y?a)}}{\vdash \text{И}x.\langle x!a\rangle, \text{Я}y.(y?a)} \quad \sim_{\mathsf{w}} \quad \text{И}_{\mathsf{load}}\,\dfrac{\text{И}_{\mathsf{pop}}\,\dfrac{\mathsf{ax}\,\dfrac{}{\vdash \langle z!a\rangle, (z?a)}}{z^{\text{И}} \vdash \langle z!a\rangle, \text{Я}y.(y?a)}}{\vdash \text{И}x.\langle x!a\rangle, \text{Я}y.(y?a)} \tag{11}$$

[7] Because of the very simple structure of terms defining atoms in PiL, only varibles can be used to substitute other varibles in terms.

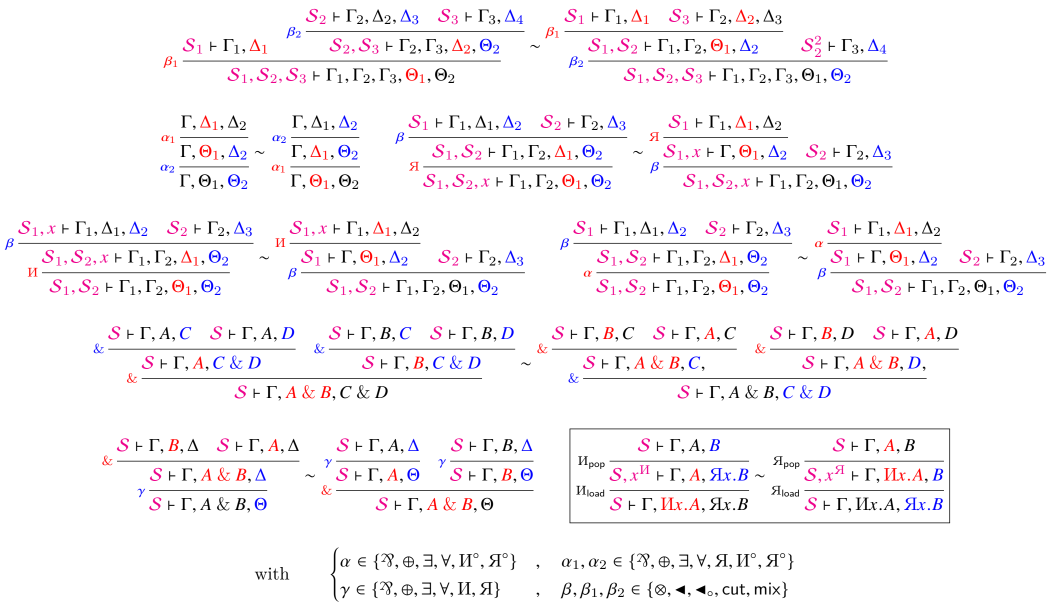
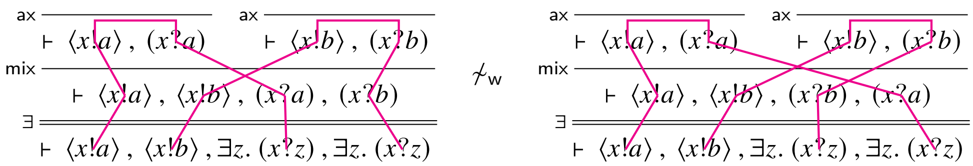


Figure 13 Local rule permutations in PiL.

In Equation (12), the existential quantifier rules select two distinct witnesses x and z, but the pair of mated atoms is the same.

$$\exists\dfrac{\mathsf{ax}\dfrac{}{\vdash \langle x!a\rangle, (x?a)}}{\vdash \exists x.\langle x!a\rangle, \exists y.(y?a)} \quad \sim_{\mathsf{w}} \quad \exists\dfrac{\mathsf{ax}\dfrac{}{\vdash \langle y!a\rangle, (y?a)}}{\vdash \exists x.\langle x!a\rangle, \exists y.(y?a)} \tag{12}$$

We could argue that in the 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. In Equation (13), we provide an example in which the choice of the witnesses for the existential quantifier rules changes the pair of atoms mated by the ax-rules.

 (13)

▶ **Theorem 27.** *Let $\mathcal{D}$ and $\mathcal{D}'$ be two derivations such that $\mathcal{D} \approx \mathcal{D}'$. Then* $\{\!\{\mathcal{D}\}\!\} = \{\!\{\mathcal{D}'\}\!\}$.

Proof. A routine induction on the size of $\mathcal{D}$ and $\mathcal{D}'$ similarly to what done in [48]. The only interesting new case is the rule permutation in the bottom-right of Figure 13 (i.e., in the box), which follows by definition of $\{\!\{\mathcal{D}\}\!\}$. ◀

▶ **Theorem 28.** *Let Λ be a proof net on Γ such that Λ sequetializes to $\mathcal{D}_\Lambda$. If $\Lambda \to^* \Lambda'$, then Λ' sequentializes to a $\mathcal{D}_{\Lambda'}$ such that $\mathcal{D}_{\Lambda'} \approx \mathcal{D}_\Lambda$.*

Proof. It suffices to check that each critical pair of coalescence steps on deductive co-trees converge. For this, we should consider the derivations labeling each link modulo the proof equivalence generated by local rule permutations Figure 13.

The most convoluted case is the one for the pair $\#/\#$, for which we use the same argument in [39], remarking that the base case of induction presents no problems in our setting (where whe 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 $\blacktriangleleft_\circ / \blacktriangleleft_\circ$. Details are provided in Appendix B. ◀

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 computation 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 [59], the absence of this property suggests that the logical framework may lack a robust design.

▶ **Notation 29.** *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 30** (Processes-as-Formulas)**.** *The formula $[\![P]\!]$ associated to a process P is inductively defined as follows:*

$$\begin{array}{lll} [\![\mathsf{Nil}]\!] = \circ \quad & [\![P \mid Q]\!]=[\![P]\!] \parr [\![Q]\!] & [\![(\nu x)(P)]\!]=\reflectbox{N}x.[\![P]\!] \\ & [\![x!\langle y\rangle.P]\!]=\langle x!y\rangle \blacktriangleleft [\![P]\!] & [\![x?(y).P]\!]=\exists y.((x?y) \blacktriangleleft [\![P]\!]) \\ \multicolumn{2}{l}{[\![x \triangleright \{\ell : P_\ell\}_{\ell\in L}]\!]=\bigoplus_{\ell\in L} ((x?\ell) \blacktriangleleft [\![P_\ell]\!])} & [\![x \triangleleft \{\ell : P_\ell\}_{\ell\in L}]\!]=\begin{cases} \langle x!\ell\rangle \blacktriangleleft [\![P_\ell]\!] & \text{if } L=\{\ell\} \\ \mathop{\&}_{\ell\in L} (\langle x!\ell\rangle \blacktriangleleft [\![P_\ell]\!]) & \text{otherwise} \end{cases} \end{array} \tag{14}$$

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

▶ **Corollary 31.** *Let P and Q be processes. If $P \equiv Q$, then $[\![Q]\!] \multimapboth [\![P]\!]$.*

▶ Remark 32. 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.

$$P = (\nu x)\,(x \triangleleft \{\ell : P_\ell\}_{\ell\in L}) \qquad \text{and} \qquad Q = x \triangleleft \{\ell : (\nu x)P_\ell\}_{\ell\in L} \tag{15}$$

These processes are not equivalent modulo the structural equivalence $\equiv$ provided in literature [80, 32], which is the same as the one we provide in Figure 2.

However, it is worth noticing that Milner's original π-calculus includes the structural equivalence in the top of Equation (16), while, at the best of our knowledge, the literature on the π-calculus does not include the corresponding structural equivalence in the bottom of Equation (16), required to capture similar interactions between choices and restriction.

$$\begin{array}{rcl} (\nu x)\,(A+B) & \equiv & (\nu x)A + (\nu x)B \\ \hline x \triangleleft \{\ell : (\nu y)P_\ell\}_{\ell\in L} & \equiv & (\nu y)\,(x \triangleleft \{\ell : P_\ell\}_{\ell\in L}) \\ x \triangleright \{\ell : (\nu y)P_\ell\}_{\ell\in L} & \equiv & (\nu y)\,(x \triangleright \{\ell : P_\ell\}_{\ell\in L}) \end{array} \tag{16}$$

This rises an interesting question on why those structural equivalences have not being used in the literature, even if the two processes in Equation (15) have the same behavior with respect to the results in type theory.

As shown in detail in [8], it is possible to associate to each computation 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 33** ([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) computation 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 14. 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 14 using rule permutations from Figure 13. 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 34. Certain works on π-calculus (e.g., [75]) 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 $a = \{\, \langle x!y\rangle\,,\, (v?w)\,\}$ of the proof net occur in the scope of a $\text{И}z$ such that $\delta_a(y) = \delta_a(w) = z$.

The canonicity result on proof nets with respect to the sequent calculus allows us to provide canonical representatives of computation 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 35.** *A **open proof nets** is a coalescent canonical co-tree Λ on a sequent Γ. The (top-down) translation in Figure 12 is extended from derivations to open derivations by translating each open premise $\mathcal{S} \vdash A_1, \ldots, A_n$ in a the link $a = \{\underbrace{A_1, \ldots, A_n}_{a}\}$ with $\delta_a = \emptyset$.*

▶ **Theorem 36.** *Two computation trees of a processes P are equivalent modulo interleaving iff they can be represented by the same open proof net.*

Proof. We associate each computation tree $\mathcal{T}$ the proof net $\{\!\{\mathcal{T}\}\!\} = \{\!\{[\![\mathcal{T}]\!]\}\!\}$ by combining the translations in Figure 14 and Figure 12. If $\mathcal{T} \sim \mathcal{T}'$, then $[\![\mathcal{T}]\!] \sim [\![\mathcal{T}']\!]$. We conclude by Theorem 28 that $\{\!\{\mathcal{T}\}\!\} = \{\!\{\mathcal{T}'\}\!\}$. ◀

▶ **Corollary 37.** *A process P is race-free iff $[\![P]\!]$ admits a unique proof net.*

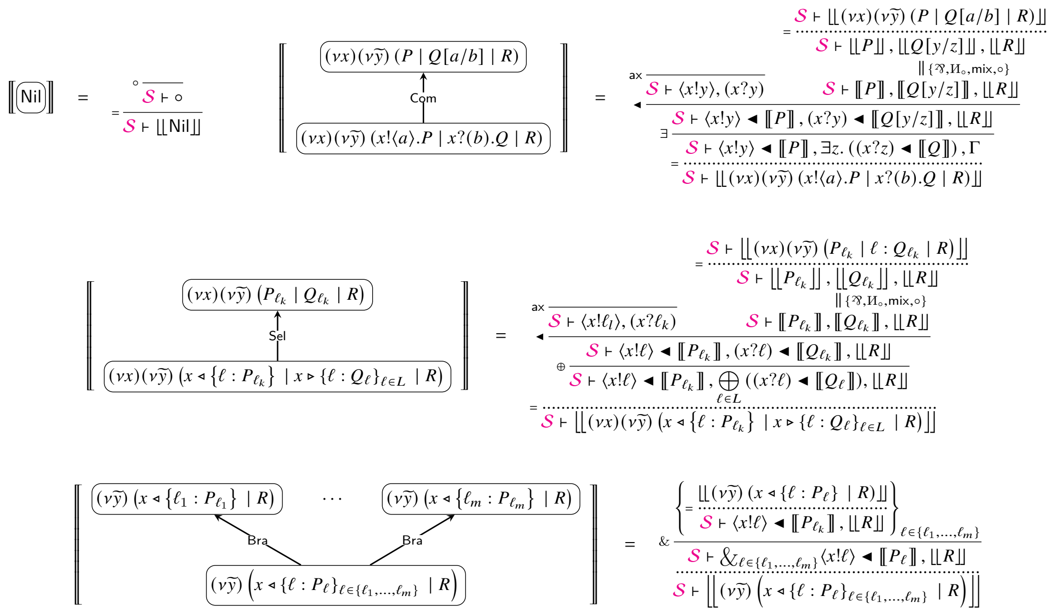


Figure 14 Translation of computation trees to derivations. If $m = 1$ in the case of Bra, then the open derivation is simply the sequent $\left\lfloor\!\!\left\lfloor (\nu\tilde{y}) \left(x \triangleleft \{\ell : P_\ell\}_{\ell\in\{\ell_1,\dots,\ell_m\}} \mid R\right) \right\rfloor\!\!\right\rfloor$.

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, 74, 31]), but rather including a non-commutative binary connective, as in Retoré's Pomset logic [70, 71] or Guglielmi's BV [37]. 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 correcntess 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.

We recalled the result from [8], where we have shown that each derivation in PiL of the formula $[\![P]\!]$ can be interpreted as a computation tree of the process P of the π-calculus (defined as in [80, 32]). We then design a novel *deduction-as-computation* paradigm in which we can interpret proof nets as canonical representatives of comptuation trees.

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. $⚑A = A \blacktriangleleft (⚑A)$) as in parsimonious linear logic [57, 4, 5] (resp. as a "parsimonious" version of the modalities from [69] and from [72]), and *recursion* using the least-fixpoint operator μ from μMALL [17, 16].

Replication	Iteration	recursion
$?\mathsf{b}\ \dfrac{\mathcal{S} \vdash \Gamma, ?A, A}{\mathcal{S} \vdash \Gamma, ?A}$	$⚑\mathsf{b}\ \dfrac{\mathcal{S} \vdash \Gamma, A \blacktriangleleft ⚑A}{\mathcal{S} \vdash \Gamma, ⚑A}$	$\mu\ \dfrac{\mathcal{S} \vdash \Gamma, A, \mu A}{\mathcal{S} \vdash \Gamma, \mu A}$

(17)

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 18 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 [70, 71], it should be possible to characterize the class of NML theorems inside Pomset, and use this logic to study sequential algorithms [69, 28, 29].

Applications to verification. This paper represent the first step in the definition of a novel paradigm allowing to intepret proofs as computation trees, and using proof equivalence to identify computation trees differing from interleaving concurrency.

Our approach could be used to develop methods complementary to the ones based on process algebras, currently lacking of way to model name mobility.[8]

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 preodreds* [27, 40, 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.

Towards a semantics for sequentiality. In this paper we used the non-associative connective $\triangleleft$ 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.

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*

[8] 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]). To the best of our knowledge, no similar investigations into nominal quantifiers have been conducted.

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, 2024. (submitted).

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

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

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

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

38 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.

39 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.

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

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

42 Ross Horne and Alwen Tiu. Towards proofs as successful executions of processes.

43 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.

44 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.

45 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.

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

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

48 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.

49 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.

50 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.

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

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

53 Olivier Laurent and Roberto Maieli. Cut elimination for monomial mall proof nets. In *2008 23rd Annual IEEE Symposium on Logic in Computer Science*, pages 486–497, 2008. doi:10.1109/LICS.2008.31.

54 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.

55 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.

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

57 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.

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

59 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.

60 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.

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

62 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`.

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

64 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`.

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

66 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`.

67 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`.

68 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`.

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

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

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

72 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`.

73 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`.

74 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`.

75 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`.

76 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`.

77 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`.

78 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`.

79 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`.

80 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`.

81 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.

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

Figure 15 Definition of $=_\alpha$ for processes.

MAV[1]-Formulas

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

Formula equivalences

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

Derivations

$$\mathcal{D} \coloneqq A \;\Big|\; \begin{array}{c} A \\ \mathcal{D}_1 \Vert \\ \mathsf{r}\,\dfrac{C'}{C} \\ \mathcal{D}_2 \Vert \\ B \end{array} \;\Big|\; \begin{array}{c} A_1 \\ \mathcal{D}_1 \Vert \\ B_1 \end{array} \odot \begin{array}{c} A_1 \\ \mathcal{D}_1 \Vert \\ B_1 \end{array} \;\Big|\; \text{Ꝺ}x.\begin{array}{c} A_1 \\ \mathcal{D}_1 \Vert \\ B_1 \end{array} \quad \text{where} \quad \begin{array}{c} \mathcal{D}_1 \Vert \\ A \end{array} \coloneqq \begin{array}{c} \circ \\ \mathcal{D}_1 \Vert \\ A \end{array}$$

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

Rules

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

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

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

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

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

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

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

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

Systems

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

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

Figure 16 Inductive definition of deep inference derivation and the rules in the system MAV[1].

A Embedding PiL into MAV[1]

We recall in Figure 16 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 18.** *Let $A_1, \ldots, A_n$ be formulas. If $\vdash_{\mathsf{PiL}} A_1, \ldots, A_n$, then $\vdash_{\mathsf{MAV}^1} \bigparr_{i=1}^n \lceil A_i \rceil$.*

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 $\bigparr_{i=1}^n \lceil A_i \rceil$ as shown in Figure 17. ◀

▶ Remark 38. 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 [78] that BV cannot have cut-free a sequent calculus, and the same holds for Retore's Pomset[9] which is a proper conservative extension of BV [67, 66].

[9] Note that a cut-free sequent calculus for Pomset has been proposed in [76], 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].

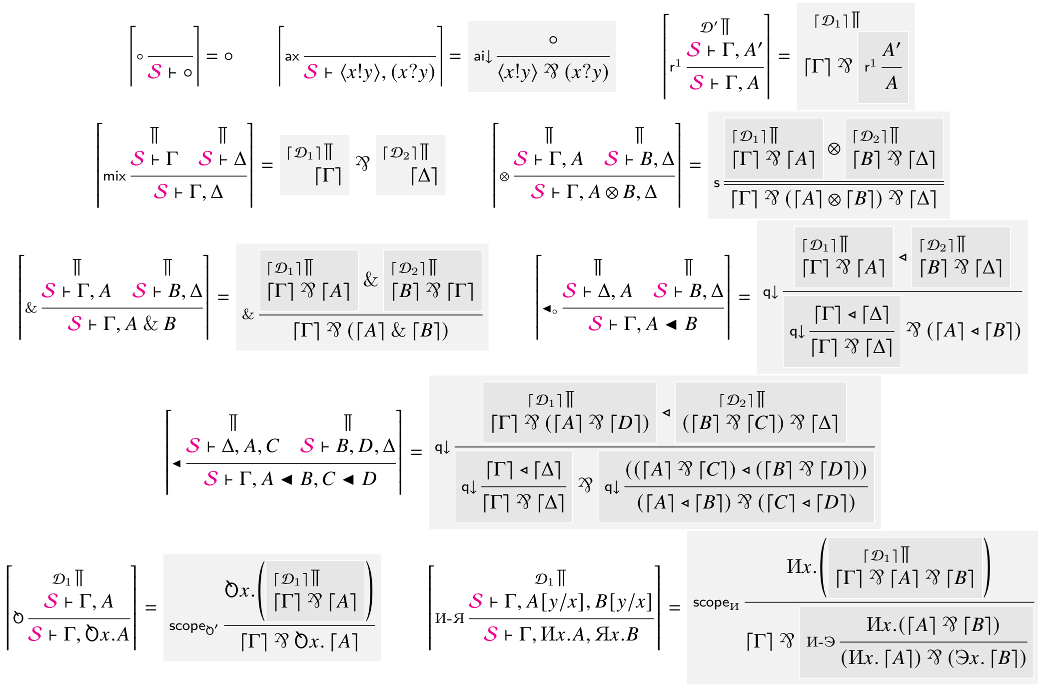


Figure 17 How to define a derivation $\lceil \mathcal{D} \rceil$ in MAV[1] from a derivation $\mathcal{D}$ in PiL, with $\mathsf{r}^1 \in \{\parr, \oplus, \exists\}$ and $\mho \in \{\forall, \text{И}, \text{Я}\}$ and $\mho' = \mho$ except if $\mho = \text{Я}$, in which case $\mho' = \ni$.

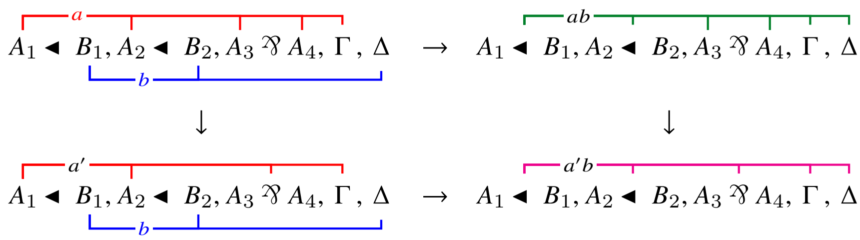
Thus we conjecture that the cause of the impossibility of having a cut-free sequent calculus for Guglielmi's BV [37] in the associativity of the connective $\triangleleft$.

B Confluence of coalescence

▶ **Theorem 28.** *Let Λ be a proof net on Γ such that Λ sequetializes to $\mathcal{D}_\Lambda$. If $\Lambda \to^* \Lambda'$, then Λ' sequentializes to a $\mathcal{D}_{\Lambda'}$ such that $\mathcal{D}_{\Lambda'} \approx \mathcal{D}_\Lambda$.*

Proof. We only discuss the critical pairs for coalescence rules not already discussed in [48]. Together with the confluence diagram of each critical pair, we show the two derivations corresponding to the two sequences of coalescence steps.

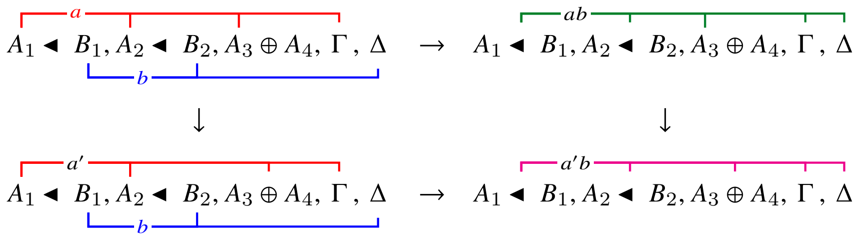
- Case $\parr/\blacktriangleleft$:



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:

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

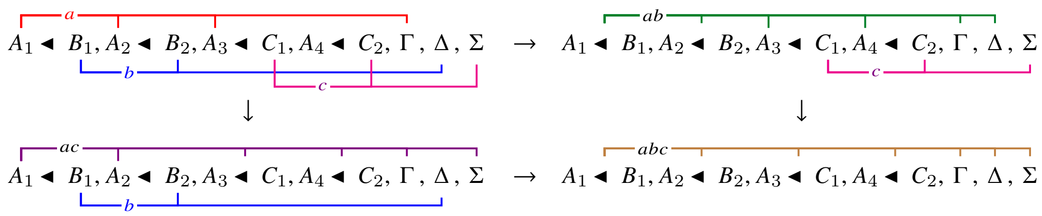
- Case $\oplus/\blacktriangleleft$:



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:

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

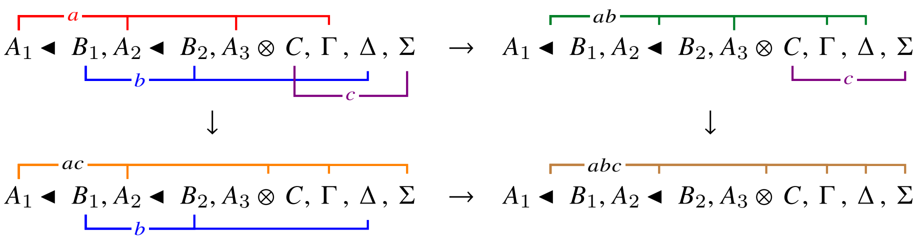
- Case $\blacktriangleleft/\blacktriangleleft$:



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:

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

- Case $\otimes/\blacktriangleleft$:

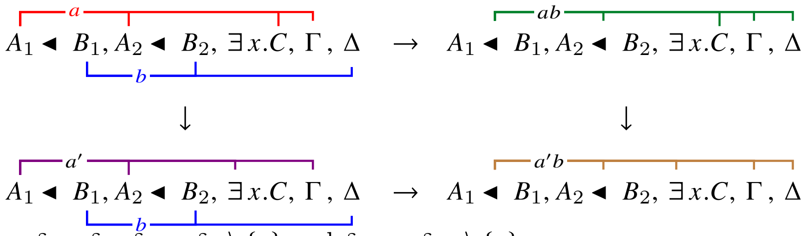


With $\delta_{ab} = \delta_a + \delta_b$, $\delta_{ac} = \delta_a + \delta_c$ and $\delta_{abc} = \delta_a + \delta_b + \delta_c$.
The two distinct derivations labeling the link in the bottom-right corner of the diagram

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

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

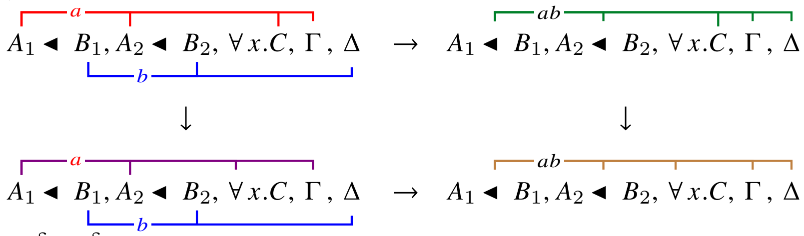
- Case $\exists/\blacktriangleleft$:



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:

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

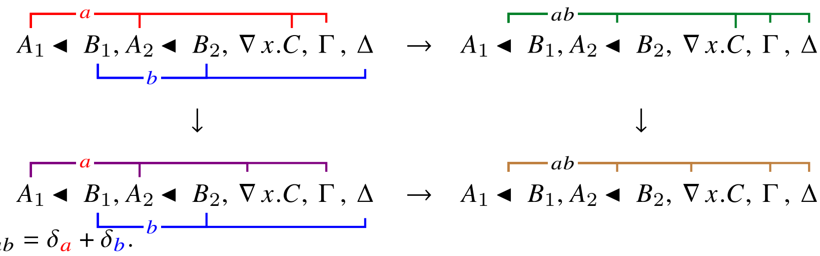
- Case $\forall/\blacktriangleleft$:



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:

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

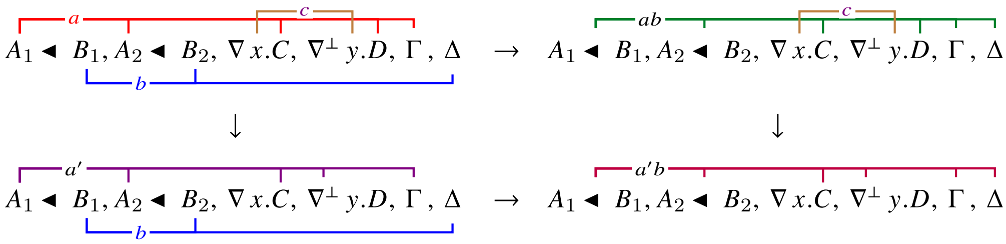
- Case $\nabla_{\mathsf{load}}/\blacktriangleleft$ with $\nabla \in \{\text{И}, \text{Я}\}$: similarly to the case $\forall/\blacktriangleleft$, but considering the rule ∇_{load} and the nominal quantifier ∇ instead of $\forall$.



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:

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

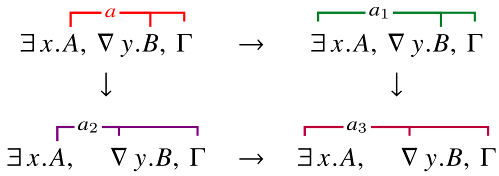
- Case $\nabla_\circ/\blacktriangleleft$ with $\nabla \in \{\text{И}, \text{Я}\}$: similar to the previous case, but considering the rule $\nabla_\circ$ instead of ∇_{load}.
- Case $\nabla_{\mathsf{pop}}/\blacktriangleleft$ with $\nabla \in \{\text{И}, \text{Я}\}$:



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:

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

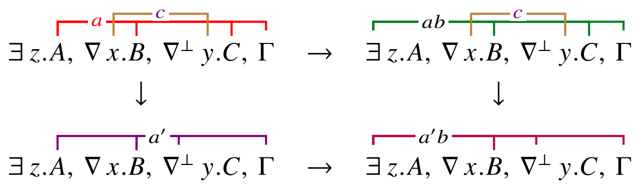
- Case $\exists/\nabla_{\mathsf{load}}$ with $\nabla \in \{\text{И}, \text{Я}\}$:



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:

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

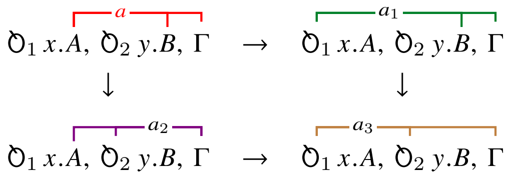
- Case $\exists/\nabla_\circ$ with $\nabla \in \{\text{И}, \text{Я}\}$: similar to the previous case, but considering the rule $\nabla_\circ$ instead of ∇_{load}.
- Case $\exists/\nabla_{\mathsf{pop}}$ with $\nabla \in \{\text{И}, \text{Я}\}$:



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:

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

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



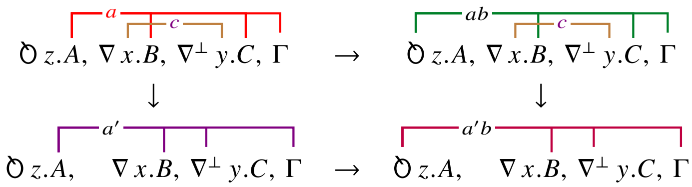
With $\delta_a = \delta_{a_i}$ for $i \in \{1,2,3\}$.
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

$$\mathsf{r}_1\dfrac{\mathsf{r}_2\dfrac{\mathcal{S} \vdash A, B, \Gamma}{\mathcal{S} \vdash A, \mho_2 y.B, \Gamma}}{\mathcal{S} \vdash \mho_1 x.A, \mho_2 y.B, \Gamma} \sim \mathsf{r}_2\dfrac{\mathsf{r}_1\dfrac{\mathcal{S} \vdash A, B, \Gamma}{\mathcal{S} \vdash \mho_1 x.A, B, \Gamma}}{\mathcal{S} \vdash \mho_1 x.A, \mho_2 y.B, \Gamma} \quad \text{if } \mathsf{r}_1, \mathsf{r}_2 \in \{\forall, И_\circ, Я_\circ\}$$

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

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

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

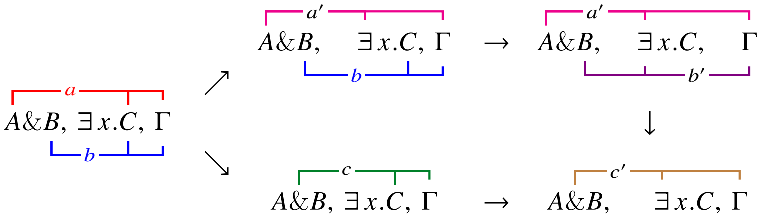


With $\delta_{ab} = \delta_a$, $\delta_{a'} = \delta_a \setminus \{y\}$ and $\delta_{a'b} = \delta_a \setminus \{y\}$
The two distinct derivations labeling the link in the bottom-right corner of the diagram according to the anticlockwise and clockwise sequence of coalescence steps are:

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

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

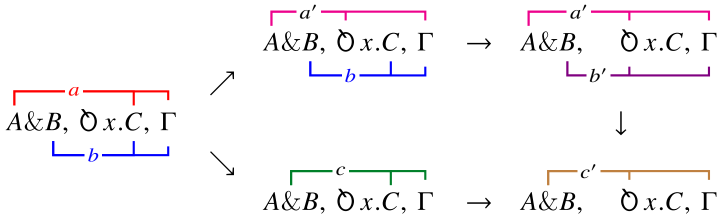
- Case &/∃:



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:

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

- Case &/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:

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

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

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

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

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.

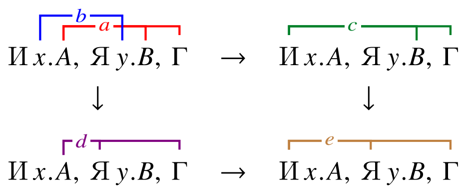
- Case mix/mix :

  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 $\invamp$. Thus, under the conditio that Λ 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 $И_{\mathsf{pop}}/Я_{\mathsf{pop}}$:



  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:

$$\mathcal{D}_1 \quad = \quad \cfrac{\cfrac{\mathcal{S} \vdash \Gamma, A, B[x/y]}{\mathcal{S}, x^{И} \vdash \Gamma, A, Яy.B}\,И_{\mathsf{pop}}}{\mathcal{S} \vdash \Gamma, Иx.A, Яy.B}\,И_{\mathsf{load}} \quad \approx \quad \cfrac{\cfrac{\mathcal{S} \vdash \Gamma, A[y/x], B}{\mathcal{S}, y^{Я} \vdash \Gamma, Иx.A, B}\,Я_{\mathsf{pop}}}{\mathcal{S} \vdash \Gamma, Иx.A, Яy.B}\,Я_{\mathsf{load}} \quad = \quad \mathcal{D}_2$$

  Where $\mathcal{D}_1 \approx \mathcal{D}_2$ because

$$\mathcal{D}_1 \sim_{\mathsf{w}} \cfrac{\cfrac{\mathcal{S} \vdash \Gamma, A[z/x], B[z/y]}{\mathcal{S}, z^{И} \vdash \Gamma, A[z/x], Яy.B}\,И_{\mathsf{pop}}}{\mathcal{S} \vdash \Gamma, Иx.A, Яy.B}\,И_{\mathsf{load}} \quad \sim \quad \cfrac{\cfrac{\mathcal{S} \vdash \Gamma, A[z/x], B[z/x]}{\mathcal{S}, z^{Я} \vdash \Gamma, Иx.A, B[z/x]}\,Я_{\mathsf{pop}}}{\mathcal{S} \vdash \Gamma, Иx.A, Яx.B}\,Я_{\mathsf{load}} \sim_{\mathsf{w}} \mathcal{D}_2$$

◀

▶ Remark 39. As already observed in [48], coalescence is not confluent on non-coalescent co-tree, as shown in the following examples.