HAL open science

# Proofs as Executions

Emmanuel Beffara, Virgile Mogbil

**To cite this version:**

Emmanuel Beffara, Virgile Mogbil. Proofs as Executions. 2011. hal-00586459v1

**HAL Id: hal-00586459**

**https://hal.science/hal-00586459v1**

Submitted on 16 Apr 2011 (v1), last revised 25 May 2012 (v2)

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Proofs as executions

**(rapport interne LIPN - Avril 2011)**

Emmanuel Beffara[1] and Virgile Mogbil[2*]

[1] IML – UMR6206, CNRS – Université Aix-Marseille 2
[2] LIPN – UMR7030, CNRS – Université Paris 13

**Abstract.** This paper proposes a new interpretation of the logical contents of programs in the context of concurrent interaction, wherein proofs correspond to successful executions of a processes. A type system based on linear logic is used, in which a given process may have several different types, each typing corresponding to a particular way of interacting with its environment, and cut elimination corresponds to executing the process in a given scenario. We prove that, for any process, the set of lock-avoiding executions precisely corresponds to the set of typings of the process. Particularly important is the role played by the axiom rule, which allows typings to make assumptions on the events of the environment. In this interpretation, logic appears as a way of making explicit the flow of causality between interacting processes.

## 1 Introduction

The extension of the familiar Curry-Howard correspondence to interactive models of computation has been an active research topic for several decades. Several systems were proposed based on linear logic [7], following the fundamental observation that it is a logic of interaction. Interpretations of proofs as processes, first formalized by Abramsky [1], later refined in particular by the first author [2], stressed that proof nets [8] and process calculi have significant similarities in dynamics. At the same time, type systems for concurrency [16] revealed to be equivalent to variants of linear logic [9]. Nevertheless, these approaches are not satisfactory yet as a logical account of concurrency, because they lack the inherent non-determinism of actual concurrency, precisely because they use logic to gain determinism.

Several approaches to the question of non-determinism in logic were also proposed. The use of the additive connectives of LL as a proof-theoretic representation of it was for instance explored by Mairson and Terui [11] to provide a notion of non-deterministic cut-elimination, or by Maurel [12] or the second author [14] who used it to represent the kind of non-determinism familiar in complexity theory. In a different style, differential logic was recently developed by Ehrhard and Regnier [6] and its untyped proof formalism was shown to represent the $\pi$-calculus faithfully [5].

* Work partially supported by the french project Complice (ANR-08-BLANC-0211-01)

The present work proposes a different approach to the topic, by questioning the "proofs-as-programs" paradigm. Proof theory wants cut elimination to be confluent, because the meaning of proofs, which lies in their normal forms. On the other hand, reduction in process calculi is execution: the final term is not the important thing, what happens to get there is the relevant information: the meaning is in the execution. Hence we propose to match proofs with executions.

Technically, we illustrate this idea in a very simple settings, in order to stress the technical novelties of the approach. Our model of interaction is finitary CCS with no choice operator (a way to treat it is sketched in the perspectives). The corresponding logic is multiplicative linear logic, with a pair of dual modalities representing actions. By representing an execution, up to unimportant interleaving aspects, as a term with linear name discipline, we get a precise correspondence between runs and proofs.

In our type system, multiplicatives represent causality and independence between parts of a run, using connectedness/acyclicity arguments to describe avoidance of deadlocks. Modalities represent explicit scheduling, using the well-known stratifying effect of boxes in proof nets. Axiom rules have an unusual interpretation: they are void of interactive content (no forwarding or copycat behaviour), but they logically implement transports of causality between different parts of a running process. Another way to interpret them is as assumptions on the available events of the environment, as we will see.

## 2 CCS runs as pairings

We consider processes of the standard language CCS [13]. The general language is defined by the following grammar. Note that we use 1 for the inactive process instead of the usual 0 because it is the neutral element of $|$ which is a multiplicative operation.

$$P, Q ::= a.P \;\mid\; \bar{a}.P \;\mid\; 1 \;\mid\; (P \mid Q) \;\mid\; P + Q \;\mid\; *P \;\mid\; (\boldsymbol{\nu} a)P$$

where $a$ is taken from an infinite set $\mathcal{N}$ of names. The main source of non-determinism in this framework is the fact that a given action name may occur several times in a given term. In order to simplify the naming of different occurrences, we decorate terms with *locations*, taken from an infinite set $\mathcal{L}$. For the purpose of the present study, we actually restrict to a very simplistic fragment. The reason for this restriction will be explained in the following development.

**Definition 1 (MCCS).** *Multiplicative CCS is the fragment of CCS that does not use choice nor replication. Actions are annotated by location, each location is used at most once in any term. Structural congruence is the smallest congruence* $\equiv$ *that makes parallel composition associative and commutative and* 1 *neutral.*

The set of locations occurring in $P$ is written $\mathcal{L}(P)$. Given $\ell \in \mathcal{L}(P)$, the *subject* of $\ell$ is the name tagged by $\ell$, written $\mathrm{subj}_P\, \ell$. The *polarity* of $\ell$ is that of the action tagged by its subject, written $\mathrm{pol}_P\, \ell$, element of $\{\pm 1\}$. Intuitively, a negative action $\bar{a}$ represents the sending of a signal on a channel $a$, and a positive action $a$ represents the reception of such a message.

**Definition 2 (execution).** *Execution is the relation over structural congruence classes, labelled by partial involutions over $\mathcal{L}$, defined by the rule*

$$\bar{a}^{\ell}.P \mid a^{m}.Q \mid R \rightarrow_{ex}^{\{(\ell,m)\}} P \mid Q \mid R$$

*Let $\rightarrow_{ex*}$ be the reflexive transitive closure of $\rightarrow_{ex}$, with the annotations defined as $P \rightarrow_{ex*}^{\emptyset} P$ and if $P \rightarrow_{ex*}^{c} Q \rightarrow_{ex*}^{d} R$ then $P \rightarrow_{ex*}^{c \cup d} R$.*

The annotation $c$ in $P \rightarrow_{ex}^{c} Q$ describes which occurrences interact in the execution step, we write $P \rightarrow_{ex} Q$ if $c$ is unimportant. Similarly, we keep locations implicit when they do not matter. Remark that, for a given $P$ and $c$, there is at most one $Q$ such that $P \rightarrow_{ex}^{c} Q$, since $c$ describes the interaction completely.

**Definition 3 (pairing).** *A* pairing *of a term $P$ is a partial involution $c$ over $\mathcal{L}(P)$ such that for all $\ell \in \operatorname{dom} c$, $\operatorname{subj} c(\ell) = \operatorname{subj} \ell$ and $\operatorname{pol} c(\ell) = -\operatorname{pol} \ell$.*

*Let $\sim_c$ be the smallest equivalence that contains $c$. Let $\leq_P$ be the partial order over $\mathcal{L}(P)$ such that $\ell <_P m$ for all subterm $x^{\ell}.Q$ of $P$ and all $m \in \mathcal{L}(Q)$. $c$ is* consistent *if $\operatorname{dom} c$ is downwards closed for $\leq_P$ and $\sim_c <_P \sim_c$ is acyclic.*

*Example 1.* The total pairings of $P = a^1.c^2 \mid b^3.\bar{a}^4 \mid \bar{b}^5.\bar{c}^6 \mid a^7.\bar{b}^8 \mid b^9 \mid \bar{a}^0$ are $c_1 = \{(9,5),(1,0),(2,6),(3,8),(4,7)\}$, $c_2 = \{(3,5),(1,4),(2,6),(7,0),(9,8)\}$, $c_3 = \{(1,4),(3,8),(7,0),(9,5),(2,6)\}$, $c_4 = \{(1,0),(3,5),(7,4),(9,8),(2,6)\}$. Only $c_1$ is inconsistent as there is a cycle induced by $\{(3,8),(4,7)\}$. The maximal consistent pairing included in $c_1$ is $\{(9,5),(1,0),(2,6)\}$.

Observe that pairings and consistency are preserved by structural congruence, as a direct consequence of the fact that subjects, polarities and prefixing are preserved by structural congruence.

**Proposition 1.** *A pairing $c$ of a term $P$ is consistent if and only if there is a term $Q$ such that $P \rightarrow_{ex*}^{c} Q$.*

*Proof (Sketch).* In an execution $P_0 \rightarrow_{ex}^{c_1} P_1 \rightarrow_{ex}^{c_2} \cdots \rightarrow_{ex}^{c_n} P_n$, the $c_i$ are disjoint, so their union is a pairing, and consistency is ensured by the fact that executions respect prefixing. Conversely, write $c = c_1 \uplus \cdots \uplus c_n$ with the $c_i$ atomic. By definition, if $c$ is consistent then $\leq_P$ induces a partial order over the domains of the $c_i$. Assume that the considered enumeration respects this order, then we can prove by recurrence that there is an execution sequence $P = P_0 \rightarrow_{ex}^{c_1} P_1 \cdots \rightarrow_{ex}^{c_n} P_n$, since each $c_i$ joins two actions of $P_{i-1}$ that are minimal for $\leq_{P_{i-1}}$.

**Proposition 2.** *Let $P$ be a term. Any two executions $P \rightarrow_{ex*}^{c} Q$ and $P \rightarrow_{ex*}^{c} R$ with the same pairing are permutations of each other, and in this case $Q \equiv R$.*

*Proof (Sketch).* Since a pairing is the union of the atomic involutions of each step, it is immediate that the execution are permutations of each other. The property $Q \equiv R$ holds easily when the executions have length one, since when reducing two independent pairs of actions, we can proceed in either order and get the same reduct in two steps. By repeated application of this basic case, we prove that the reducts are equal by showing that one execution can be transformed into the other by successive exchanges of consecutive independent steps.

We will thus consider consistent pairings as the proper notion of execution for CCS terms. Maximal consistent pairings represent executions of processes until a state where no more execution is possible. A useful tool in the study of pairings is the following notion of determinisation, by which we can turn a pairing of a term into a term that has no other pairing. In other words, determinisation is a way to represent a run of a term in the language of MCCS itself.

**Definition 4 (deterministic term).** *A term $P$ is* deterministic *if it has at most one occurrence of each action.*

The pairings of a deterministic term form a lattice, consistent pairings too, so there is a unique maximal consistent pairing for any deterministic term.

**Definition 5 (determinisation).** *Assume an injective map $\delta : \mathcal{N} \times \{\pm 1\} \times \mathcal{L} \to \mathcal{N}$. Given a term $P$ and a partial involution $c$, define the* determinisation *$\lfloor P \rfloor_c$ of $P$ along $c$ inductively as $\lfloor 1 \rfloor_c = 1$, $\lfloor P \mid Q \rfloor_c = \lfloor P \rfloor_c \mid \lfloor Q \rfloor_c$ and*

$$\lfloor a^\ell.P \rfloor_c = \delta(a,+1,\ell)^\ell.\lfloor P \rfloor_c\,, \quad \lfloor \bar{a}^\ell.P \rfloor_c = \begin{cases} \overline{\delta(a,+1,\ell)}^\ell.\lfloor P \rfloor_c & \text{if } \ell \in \operatorname{dom} c, \\ \delta(a,-1,\ell)^\ell.\lfloor P \rfloor_c & \text{otherwise.} \end{cases}$$

By construction, $\lfloor P \rfloor_c$ is deterministic, the pairings of $\lfloor P \rfloor_c$ are the restrictions of $c$, consistency preserved, so $c$ is the unique maximal pairing of $\lfloor P \rfloor_c$.

*Example 2.* For the term $P = a^1.c^2 \mid b^3.\bar{a}^4 \mid \bar{b}^5.\bar{c}^6 \mid a^7.\bar{b}^8 \mid b^9 \mid \bar{a}^0$ and pairings of example 1, we obtain the following determinisations.
$c_3 = \{(1,4),(3,8),(7,0),(9,5),(2,6)\}$ induces $\lfloor P \rfloor_{c_3} = a.c \mid b.\bar{a} \mid \bar{e}.\bar{c} \mid d.\bar{b} \mid e \mid \bar{d}$,
$c_4 = \{(1,0),(3,5),(7,4),(9,8),(2,6)\}$ induces $\lfloor P \rfloor_{c_4} = a.c \mid b.\bar{d} \mid \bar{b}.\bar{c} \mid d.\bar{e} \mid e \mid \bar{a}$.

If we extended our study to the whole of CCS, determinisations would still be in MCCS. The theory of pairings would have to be refined: external choice requires a notion of conflict in the space of locations (as in event structures [15]), replications requires the introduction of indices to distinguish copies. The restriction operator $(\boldsymbol{\nu} a)$ serves two purposes: it limits the scope of a name, and it makes it possible to have names local to each copy in a replication; both these features are irrelevant in the deterministic case, hence we leave it out of MCCS.

## 3 MLL with linear modalities

We now present the logic we use to describe pairings of MCCS. For similar reasons as in CCS, we use a very simple system, namely the multiplicative fragment of linear logic [4], augmented with a pair of dual modalities that describe actions.

**Definition 6 (M?LL).** *The formulas of M?LL are built by the grammar*

$$A, B ::= \alpha \;\mid\; \alpha^\perp \;\mid\; A \otimes B \;\mid\; A \parr B \;\mid\; ?_a A \;\mid\; ¿_a A$$

*where the $\alpha$ are literals and the $a$ are names in $\mathcal{N}$. The* negation *$A^\perp$ of a non-literal formula $A$ is defined by de Morgan duality as $(A \otimes B)^\perp = A^\perp \parr B^\perp$ and $(?_a A)^\perp = ¿_a A^\perp$. A* type *$(\Gamma, \Delta \ldots)$ is a finite multiset of formulas. Type derivations are built from the rules of table 1, ignoring the left side of $\vdash$.*

Axiom and cut rules ($A$ is a M?LL formula):

$$\frac{}{1 \vdash A, A^{\perp}}\ (\mathrm{ax}) \qquad\qquad \frac{P \vdash \Gamma, A \qquad Q \vdash A^{\perp}, \Delta}{P \mid Q \vdash \Gamma, \Delta}\ (\mathrm{cut})$$

Multiplicative rules:

$$\frac{P \vdash \Gamma, A, B}{P \vdash \Gamma, A \parr B}\ (\parr) \qquad\qquad \frac{P \vdash \Gamma, A \quad Q \vdash B, \Delta}{P \mid Q \vdash \Gamma, A \otimes B, \Delta}\ (\otimes)$$

Modality rules (dereliction and codereliction):

$$\frac{P \vdash \Gamma, A}{a^{\ell}.P \vdash \Gamma, ?_a A}\ (?) \qquad\qquad \frac{P \vdash \Gamma, A}{\bar{a}^{\ell}.P \vdash \Gamma, ¿_a A}\ (co?)$$

**Table 1.** Inference rules in M?LL

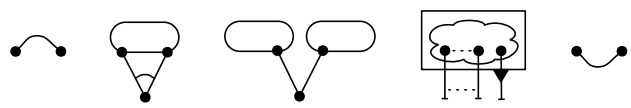


**Fig. 1.** Representation of proof structures: axiom link, $\parr$ node, $\otimes$ node, boxes, cut.

Proofs in sequent calculus are well suited to inductive reasoning, however their use on proof theory is uneasy because their rigid structure obscures many reasonings, like those below in particular. For this reason, we will turn to proof nets, using the standard machinery of linear logic [8, 4]. Modality rules are represented using boxes (like promotions in standard linear logic, but with different typing rules). The only extra information we add to standard proof structures is the location of each box, to reflect the use of locations in CCS terms in the sequel. For those who are not familiar with the standard definitions of proof structures and proof nets, they are put in appendices. We detail here specificities of M?LL.

**Definition 7 (proof structure).** *A* proof structure *consists of an ordered forest of nodes labelled by formulas, denoted $x^A$, together with a set $Ax$ of axiom links (pair of leaves), a set $Cut$ of cuts (pair of roots) and a set $Box$ of modality boxes such that each box $\beta$ has a unique location $\ell(\beta)$.*

Each box $\beta$ is associated (inductively) to a proof structure $S$ whose conclusions are in bijection with the ports of the box. The *principal port* $x^A$ of $\beta$ for $A = ?_a B$ or $A = ¿_a B$ is in bijection with a conclusion of $S$ labeled $B$, whereas it is the same label for the *auxiliary ports*.

The roots that are not part of a cut are called the *conclusion nodes* of $S$. The *conclusion type* of $S$ is the multiset of the labels of its conclusion nodes.

The graphical notation of proof structures is presented in figure 1. By definition there are arcs only to multiplicative nodes, moreover proof structures can be drawn considering the top-bottom orientation of arcs, so we keep arc orientation

implicit by this convention. Arcs to a $\parr$ node are joint by a circle on the side of this node. By construction, the conclusion labels suffice to deduce all labels, so we keep most of this information implicit.

**Definition 8 (proof net).** *A* proof net *is a proof structure built following M?LL sequent calculus rules. An* immediate subnet *of a proof net $\pi$ is an induced subgraph of $\pi$ that is itself a proof net. A* subnet *of $\pi$ is either an immediate subnet of $\pi$ or (inductively) a subnet of a box of $\pi$.*

Well known correctness criteria [4, 8] apply to characterise proof nets among proof structures by combinatorial means like acyclicity and connectedness.

**Definition 9 (cut elimination).** *Let $\pi$ and $\pi'$ be a proof structure and $c$ be a partial involution on $\mathcal{L}$. We have $\pi \to^c_{ce} \pi'$ if $\pi$ contains a cut $\kappa = \{x, y\}$ either at top level or inside a box and one of the following cases occurs:*

- *Multiplicative step and Axiom step: as usual but $c = \emptyset$.*
- *Modality step: If $x$ and $y$ are principal ports of two boxes $\beta, \beta'$, then $c$ permutes $\ell(\beta)$ and $\ell(\beta')$ and $\pi'$ is obtained by replacing each box with its associated proof structure.*
- *Commutation step (unused here): If $x$ is the auxiliary port of a box $\beta$, then $c = \emptyset$, and the cut and a subnet of $\pi$ that contains $y$ are moved inside $\beta$.*

*The* annotated cut elimination *relation $\to^c_{ce}$ over proof structures is the reflexive transitive closure of the rules above (if $\pi \to^c_{ce} \pi' \to^d_{ce} \pi''$ then $\pi \to^{c \cup d}_{ce} \pi''$).*

Our proof system enjoys a standard cut-elimination theorem using this definition: if $\pi \to^c_{ce} \pi'$ and $\pi$ is a proof net, then $\pi'$ is a proof net with the same conclusion (this is proved by standard arguments using correctness criteria, hence we will not develop this point); if a proof $\pi$ is irreducible by $\to_{ce}$, then it has no cut link (this is an immediate case analysis). Note however that $\to_{ce}$ is not confluent, because of commutation steps.

**Definition 10 (head reduction).** *Head reduction is the annotated relation $\to^c_h$ over proof structures defined as the restriction of $\to^c_{ce}$ that only applies at top level and does not use the commutation step of cut elimination.*

This particular strategy is relevant because it does not reduce inside boxes, that is under prefixes, it only affects cuts in active position (from the point of view of processes). However, this strategy does not eliminate all cuts in general.

In the analysis of proofs, the following notion of path will be useful. It describes a way to traverse arcs and axioms/cuts in a proof structure while respecting the logical meaning of formulas.

**Definition 11 (path).** *A* path *in a proof structure $S$ is an alternating path in the underlying graph of $S$, such that alternations occur only at axioms, cuts and boxes. Each move between ports $x$ and $y$ of a box $\beta$ must be associated with a path between the corresponding conclusions in $\beta$. We further require a typing constraint: a path can only move up a left (resp. branch) if has moved down a left (resp. right) branch before, with a natural well-bracketing condition.*

For instance, a path starting from an axiom with type $\alpha$ may move down the tree of nodes, reach a cut, move up the other side of the cut, always in the branches that contain $\alpha$, reach an axiom, and so on.

## 4 Typing MCCS terms

Proofs in M?LL will serve as a type system. Although this can be formulated in usual sequent style (as in table 1), the natural notion rather relates proof nets and structural congruence classes of terms.

**Definition 12 (term assignment).** *Let $S$ be a proof structure. The MCCS term $\lfloor S \rfloor$ assigned to $\pi$ is the parallel composition of the $\lfloor \beta \rfloor$ for each box $\beta$ in $S$. In turn, for a box $\beta$ with location $\ell$ and associated structure $S_\beta$, the term $\lfloor \beta \rfloor$ is $a^\ell.\lfloor S_\beta \rfloor$ if the principal port of $\beta$ has modality $?_a$ and $\bar{a}^\ell.\lfloor S_\beta \rfloor$ if the principal port of $\beta$ has modality $¿_a$. A term $P$ is said to have type $\Gamma$ if there is a proof net $\pi$ of conclusion $\Gamma$ such that $\lfloor \pi \rfloor \equiv P$. In this case we write $\pi : P \vdash \Gamma$.*

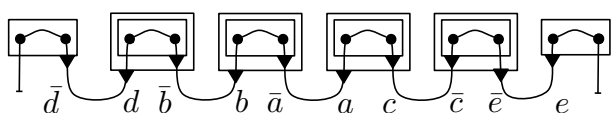
A proof net is a proof structure that is built using the rules of table 1, ignoring the terms on the left of the $\vdash$ symbols. It is obvious that these terms do reflect the definition of term assignment: A term $P$ has type $\Gamma$ if and only if there is a type derivation with conclusion $P \vdash \Gamma$ using the rules of table 1.

We now establish the correspondence between cut elimination in a proof and execution steps in the assigned terms. The first result justifies head reduction:

**Proposition 3.** *Let $\pi$ be a proof structure. For all head reduction $\pi \to_h^c \pi'$ there is an execution $\lfloor \pi \rfloor \to_{ex*}^c \lfloor \pi' \rfloor$.*

*Proof (Sketch).* Axiom and multiplicative cut eliminiation steps do not affect the assigned terms, besides their annotation is empty, so the result holds immediately for them. When a modality step applies, it reduces a cut between boxes with dual modalities (because of typing), hence the associated terms are ready to interact; the reduct is easily seen to be the assigned term of the reduct proof.

*Example 3.* Let $\pi$ be the following proof net.




We have $\lfloor \pi \rfloor = a.c \mid b.\bar{a} \mid \bar{e}.\bar{c} \mid d.\bar{b} \mid e \mid \bar{d}$. (It is $\lfloor P \rfloor_{c_3}$ of previous examples). As it is deterministic term, we abusively merge locations with names. We consider the head reduction sequence $\pi \to_h^z \pi'$ (where $\pi'$ is an axiom link) for $z = \{(d, \bar{d}), (b, \bar{b}), (a, \bar{a}), (e, \bar{e}), (c, \bar{c})\}$. We have $\lfloor \pi \rfloor \to_{ex*}^z \lfloor \pi' \rfloor \equiv 1$.

Subject reduction does not hold in general, however. Indeed, a given proof may hold several occurrences of a given modality, corresponding to different occurrences of an action in the term, and the structure of cuts may not match a given execution step. This is not a defect, since we actually intend to type pairings rather than processes: we do get subject reduction if we restrict to proofs that describe deterministic terms.

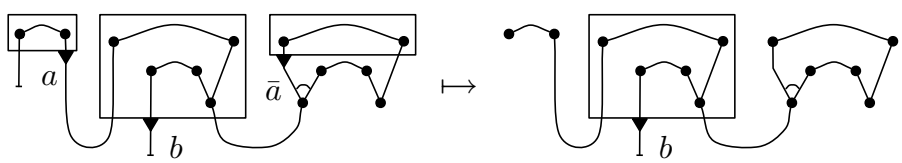
**Definition 13 (linear proof).** *A proof structure $S$ is called* linear *if*
- *$S$ contains at most one box for each modality,*
- *all occurences of a given modality $¿_a A$ in the labels in $S$ have the same immediate subformula $A$, and if $¿_a A$ and $?_a B$ occur then $A$ and $B$ are dual,*
- *if $S$ contains a box for both $?_a A$ and $¿_a A^\perp$, then neither formula occurs in the conclusion type of $S$.*

The essence of the linearity condition is the first constraint. Intuitively, the second and third constraints serve to guarantee that the property is preserved by composition. Indeed, if a formula $?_a A$ occurs in the conclusion of a proof $\pi$, then the proof may be cut against a proof that contains a modality box for $¿_a A^\perp$, which breaks linearity if $\pi$ already contains a box for some $¿_a B$. Note that the fact of being a linear proof is preserved by cut elimination.

**Theorem 1 (subject reduction).** *Let $P$ be a term of type $\Gamma$, typed by a linear proof $\pi$. For all execution $P \to^c_{ex*} P'$ there is a linear proof $\pi' : P' \vdash \Gamma$.*

*Proof (Sketch).* An execution step $\lfloor \pi \rfloor \to^{(\ell,m)}_{ex} P$ involves immediate subterms $a^\ell.Q$ and $\bar{a}^m.R$ for $a \in \mathcal{N}$. Then $\pi$ contains two top level boxes with respective principal ports $x^{?_a A}$ and $y^{¿_a A^\perp}$, for $A \in M?LL$. Since $\pi$ is linear, $x$ and $y$ are elimination boxes for each other, ending a path $\rho$ (as of definition 11) whose axioms contain modalities of $x$ and $y$ in their types. Let $\pi'$ be the rewriting of $\pi$ where such modalities are removed (boxes are rewrited by their contents, axioms by axioms on $A/A^\perp$). Clearly $\pi'$ is a linear proof that infers $P$.

This theorem states that types are preserved by execution. However, the proof uses a rewriting of the typing proofs that does not correspond to cut elimination in general. Indeed, consider the following example of typing, called $\pi$ the l.h.s.:




Then the proof is linear, irreducible by head cut elimination, but the assigned term $\lfloor \pi \rfloor = \bar{a} \mid \bar{b} \mid a$ does execute into $\bar{b}$. In $\pi$, this involves a cut on the axiom inside the middle box. As done in theorem 1 the rewriting of $\pi$ in a linear proof $\pi'$ assigned to $\bar{b}$ is the r.h.s..

We can get a precise correspondence between execution and head cut elimination by imposing an additional constraint on the shape of proofs.

**Definition 14 (regular proof).** *Let $\pi$ be a proof structure. A* transport axiom *is an axiom link contained in box $\beta$ with modalities in its type. Such an axiom is called* anchored *if there is a path from one of its conclusions to an auxilary port of $\beta$ and one from its other conclusion to the principal port. A box is* simple *if it contains no axiom introducing modalities. A proof structure $\pi$ is called* regular *if all transport axioms in $\pi$ are anchored and all codereliction boxes are simple.*

**Theorem 2 (strong subject reduction).** *Let $\pi$ be a regular linear proof net. For all execution $\lfloor\pi\rfloor \to^{c}_{ex*} P$ there is a regular linear proof $\pi'$ such that $\pi \to^{c}_{h} \pi'$ and $\lfloor\pi'\rfloor = P$.*

*Proof (Sketch).* Consider an execution step $\lfloor\pi\rfloor \to^{(\ell,m)}_{ex} P$. As in theorem 1 proof, linearity implies there is boxes at top level and a path $\rho$ between their principal ports $x^{?_a A}$ and $y^{¿_a A^\perp}$ for immediate subterms $a^\ell.Q$ and $\bar{a}^m.R$ of $\lfloor\pi\rfloor$. Since $x$ is cut at top level, there is no box traversal along $\rho$. Otherwise linearity or regularity is contradicted. Then $\rho$ is a multiplicative cut path whose cut elimination $\to^{\emptyset}_{h}$ until $x$ and $y$ preserves $\lfloor\pi\rfloor$ as well as regularity and linearity of proof.

## 5 Divisions of proof nets

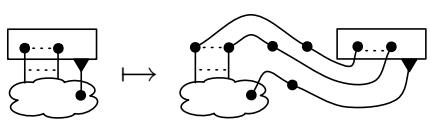
The point of this section is to provide a kind of reciprocal statement for subject reduction: if a term $T$ can reduce into a typed term $T'$, then we can type $T$ with a proof that reduces to the typing of $T'$. Because we want logically correct proof structures, this operation requires some care.

*Example 4.* Consider the term $P := a.\bar{b} \mid b.\bar{c} \mid \bar{a}.c$. We cannot simply type each thread with a simple type like $?_a\alpha, ¿_b\alpha^\perp$ and then introduce a cut for each interaction, since we would get a cyclic structure.

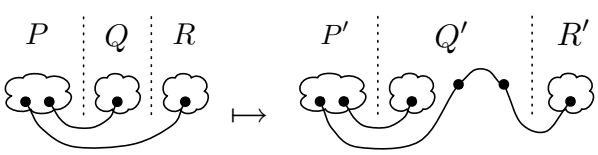
We now describe a general method for deducing a typing by "anti-execution" of a proof. We stay at a partly informal level for clarity, all formal statements are detailed in the appendix.

Consider a generic execution step $P \mid a.Q \mid \bar{a}.R \to_{ex} P \mid Q \mid R$. Assume the reduct is typed by some proof $\pi$. We want to put the parts of $\pi$ that correspond to $Q$ and $R$ into boxes, with a cut between them, while rewriting the proof to avoid cycles. For this purpose, we proceed in four steps:

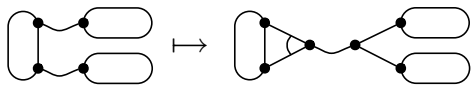
**Selection** consists in moving each box belonging to $Q$ or $R$ away from the main proof, by means of an axiom/cut pair, so that $Q$ and $R$ are represented by simple sets of boxes, cut with the main proof (which corresponds to $P$), with no multiplicative connectives:



**Chaining** consists in introducing an extra axiom/cut pair in the middle of each cut between $P$ and $R$, so that there are cuts only between $P$ and $Q$ or $Q$ and $R$, and not between $P$ and $R$ directly:



**Simplification** consists in making sure that there is actually exactly one cut between $P$ and $Q$ and one between $Q$ and $R$, by multiplexing multiple cuts through multiplicatives:



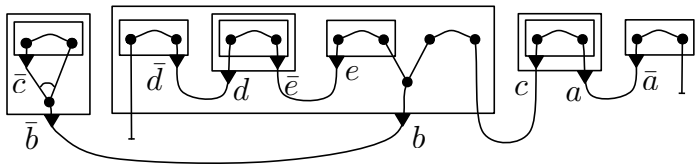
Correctness criteria guarantee that we can always find two cuts for which there is one connected component on one side, two on the other.

**Boxing** consists in putting $Q$ and $R$ into boxes, cut together, so that $Q$ has one auxiliary port to $P$ and $R$ has no auxiliary port:

Following this method, we prove the following statement:

**Proposition 4 (anti-execution).** *Let $T_1 \rightarrow^c_{ex} T_2$ be an execution step and let $\pi_2 : T_2 \vdash \Gamma$ be a typing. There exists a typing $\pi_1 : T_1 \vdash \Gamma$ such that $\pi_1 \rightarrow^c_h \pi_2$.*

*Example 5.* Consider the term of $P$ of example 1. We consider the execution $e = (a, \bar{a})(b, \bar{b})(c, \bar{c})(d, \bar{d})(e, \bar{e})$ of the determinized term $\lfloor P \rfloor_{c_4} = a.c|b.\bar{d}|\bar{b}.\bar{c}|d.\bar{e}|e|\bar{a}$ for the (total and consitent) pairing $c_4 = \{(1, 0), (3, 5), (7, 4), (9, 8), (2, 6)\}$. A typing synthetized by the construction of proposition 4 is the following.




**Lemma 1 (preserved regularity).** *In the construction of proposition 4, if $\pi_2$ is regular, then so is $\pi_1$. If $\pi_2$ is linear and $T$ is deterministic, then $\pi_1$ is linear.*

*Proof.* In our construction, i) all axioms added by selection are in $p$ and eventually in $p''$, ii) all axioms added by chaining are in $q'$ and eventually in $q''$ by simplification, iii) no axioms are added in $r''$. The last boxing step ensures that all axioms added in $q''$ are transport axioms which are anchored, and the codereliction box is simple, so the regularity is preserved. Linearity is preserved when we add new axioms on modalities and new cuts on it, it is always for already existing modalities, up to the last added boxes introducing also news modalities.

*Example 6.* In the previous example 5, one can also start execution by $(b, \bar{b})(a, \bar{a})$ as seen in the typing. All execution permutation of $\lfloor P \rfloor_{c_4}$ in the pairing $c_4$ is allowed by the typing proof synthetized from the execution $e$.

## 6 Correspondence

We now summarize the previous results, about subject reduction and the reverse operation, into a precise statement relating typings and execution.

**Lemma 2 (initial typing).** *Every MCCS term is typable by a cut-free regular proof with at least one non-modal conclusion.*

*Proof.* We simply build a proof of $T \vdash A_T, B_T$ with $A_T$ non-modal by induction on $T$. For $T = 1$, use the axiom rule to get $1 \vdash \alpha^\perp, \alpha$. For $T = P \mid Q$, deduce $T \vdash A_P \parr A_Q, B_P \otimes B_Q$ by the tensor rule. For $T = a.P$, deduce $T \vdash A_P, ?_a B_P$ by the action rule, and similarly for co-actions. The proof thus built is obviously regular since it has no transport axiom.

**Theorem 3.** *For every execution $P \to^c_{ex*} Q$ there are typings $\pi_P : P \vdash \Gamma$ and $\pi_Q : Q \vdash \Gamma$ such that $\pi_P \to^c_h \pi_Q$. Moreover, for every execution sequence $P \to^{c_1}_{ex} P_1 \cdots \to^{c_n}_{ex} P_n = Q$ with $c_1 \cup \cdots \cup c_n = c$, there is a cut elimination sequence $\pi_P \to^{c_1}_h \pi_1 \cdots \to^{c_n}_h \pi_n = \pi_Q$, with $\lfloor \pi_i \rfloor = P_i$ for all $i$.*

*Proof.* By lemma 2 we can find a regular cut-free proof $\pi_Q : Q \vdash \Gamma$ such that some formula in $\Gamma$ is non-modal. If we apply proposition 4 repeatedly to $\pi_Q$ with the steps of the considered execution $P \to^c_{ex*} Q$, we get a proof $\pi_P : P \vdash \Gamma$ that reduces to $\pi_Q$ by a head reduction sequence labelled $c$. Clearly, the types in $\pi_P$ can be consistently rewritten to get a type derivation $\pi'_P$ for the determinisation $\lfloor P \rfloor_c$, and $\pi'_P$ will be linear and regular. Moreover, every execution sequence of $P$ with label $c$ will be an execution sequence of $\lfloor P \rfloor_c$ with the same label. By lemma 1, $\pi'_P$ enjoys strong subject reduction as of theorem 2, hence every run of $\lfloor P \rfloor_c$ labelled by $c$ corresponds to a head reduction sequence in $\pi'_P$ labelled by $c$. The reduct of $\lfloor P \rfloor_c$ by this sequence is $\lfloor Q \rfloor_c$ and the reduced proof $\pi'_Q$ is such that $\lfloor \pi'_Q \rfloor = \lfloor Q \rfloor_c$. By applying the relabelling used for detrminisation in reverse, we get a head reduction sequence from $\pi_P$ to $\pi_Q$ with the same labels.

In other words, every execution of a term can be exactly characterized up to permutation by typing, in the sense that the execution sequences of the term within the same pairing will be exactly the head reduction sequences of the associated typing proof. By combining determinisation (definition 5) and strong subject reduction (theorem 2) we get that, conversely, each regular typing of a term defines a set of executions stable by permutation.

Lemma 2 seems to state that every term is typable, hence it is legitimate to ask what our system guarantees. However, one should remeber that, although the typing rules of table 1 apply to terms, what we intent to type is actually executions. Considering the set of admissible types for a given execution, we can indeed characterize when executions can be composed.

## 7 Conclusion and further works

In this work we have developed, in the simple framework of multiplicative CCS, a precise logical description of executions of processes. A key technical tool is

the use of pairings, by which we separate non-determinism in communication from the multiplicity of equivalent schedulings; this technique extends well to more expressive frameworks (full CCS, $\pi$-calculs, etc.). The logical interpretation we propose moves beyond the traditionnal Curry-Howard for concurrency by accepting non-deterministic terms, albeit with a change of interpretation in the correspondence. Indeed, the logic we use is well studied and has a wide range of existing tools (efficient correctness criteria, proof search, etc.) but its interpretation in our paradigm of proof-as-executions is new.

*Logical expressiveness* The restriction to purely mutliplicative objects, in MCCS and MLL, lets us concentrate on the precise role of multiplicatives and axioms as descriptions of how a process interacts with its environment, however it hides the complexity inherent to the other defining features of concurrent systems: choice and recursion.

In the first case, our system can extend rather naturally. The technique of pairings still works, consistency simply needs to take into account a notion of conflict as in event structures [15]. The type system is naturally extended by additive rules:

$$\frac{P \vdash \Gamma, A \quad Q \vdash \Gamma, B}{P + Q \vdash \Gamma, A \,\&\, B} \qquad \frac{P \vdash \Gamma, A}{P \vdash \Gamma, A \oplus B}$$

possibly with the restriction that $A$ and $B$ are modal. Then we can type useful processes that use choice. For instance, describe a boolean on names $t, f$ as some process that will send a signal on one of the channels $t, f$. This can be materialized by the type $B(t, f) := \alpha^\perp, ¿_t\alpha \otimes ¿_f\alpha$ which reads like "give me control (using $\alpha$), I will terminate by a signal on $t$ or $f$". Then consider a negation function: $N := t.\bar{f}' + f.\bar{t}'$. By studying its interactions with the environments $E_1 := \bar{t} \mid f'$ and $E_2 := \bar{f} \mid t'$, we see that both $N \mid E_i$ have complete consistent pairings, hence we can type $\alpha^\perp, \alpha$. Extracting the types of $N$ we get $?_t\alpha^\perp, ¿_{f'}\alpha$ and $?_f\alpha^\perp, ¿_{t'}\alpha$, which we combine by additives into a unique type $?_t\alpha^\perp \,\&\, ?_f\alpha^\perp, ¿_{t'}\alpha \oplus ¿_{f'}\alpha$. This way we get a possible specification for $N$.

Using similar arguments, in the presence of replication, we can combine several executions differing by the number of copies of a server used by using exponential modalities.

*Causality* A crucial feature of our work is the intepretation of axioms as a way to transfer causality. An effect is that, most of the time, the type of a term will contain modalities for actions that it does not contain by itself. For instance, $a.\bar{b}$ may have type $?_a?_c\alpha^\perp, ¿_b¿_c\alpha$, which can be read "give me a signal on $a$ with the promise of a signal on $c$, and I will answer with a signal on $b$ and the promise of a signal $c$". This makes it explicit that this part of interaction will be involved in the triggering of interaction on $c$, but only indirectly by allowing bearers of $c$ to get active. This idea suggests new ways of analysing causality in interactive systems, and the fact that the flow of causality is often as complicated as the flow of information. Besides, a similar fact is illustrated by the expressiveness of solos [10, 3], where communication is used to carry all prefixing information in processes. Our interpretation may provide a logical insight on this matter.

## References


1. Samson Abramsky. Proofs as processes. *Theor. Comput. Sci.*, 135(1):5–9, 1994.
2. Emmanuel Beffara. A concurrent model for linear logic. In *21st International Conference on Mathematical Foundations of Programming Semantics (MFPS)*, volume 155, pages 147–168, may 2006.
3. Emmanuel Beffara and François Maurel. Concurrent nets: a study of prefixing in process calculi. *Theoretical Computer Science*, 356(3):356–373, may 2006.
4. Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28(3):181–203, 1989.
5. Thomas Ehrhard and Olivier Laurent. Interpreting a finitary $\pi$-calculus in differential interaction nets. In Luís Caires and Vasco T. Vasconcelos, editors, *18th International Conference on Concurrency Theory (Concur)*, volume 4703 of *LNCS*, pages 333–348. Springer, September 2007.
6. Thomas Ehrhard and Laurent Regnier. Differential interaction nets. In *Workshop on Logic, Language, Information and Computation*, 2004. Invited paper.
7. Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50(1):1–102, 1987.
8. Jean-Yves Girard. Proof-nets : the parallel syntax for proof theory. *Logic and Algebra*, 180, 1996.
9. Kohei Honda and Olivier Laurent. An exact correspondence between a typed $\pi$-calculus and polarised proof-nets. *Theoretical computer science*, 411(22–24):2223–2238, May 2010.
10. Cosimo Laneve and Björn Victor. Solos in concert. In JiŘí Wiederman, Peter van Emde Boas, and Mogens Nielsen, editors, *26th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1644, pages 513–523. Springer Verlag, jul 1999.
11. Harry G. Mairson and Kazushige Terui. On the computational complexity of cut-elimination in linear logic. In Carlo Blundo and Cosimo Laneve, editors, *Theoretical Computer Science, 8th Italian Conference, ICTCS*, volume 2841 of *Lecture Notes in Computer Science*, pages 23–36. Springer, 2003.
12. François Maurel. Nondeterministic light logics and NP time. In Martin Hofmann, editor, *Typed Lambda Calculi and Applications (TLCA), 6th International Conference*, number 2701 in LNCS, pages 241–255. Springer, 2003.
13. Robin Milner. *Communication and concurrency*. Prentice Hall, 1989.
14. Virgile Mogbil. Non-deterministic boolean proof nets. In Marko van Eekelen and Olha Shkaravska, editors, *Foundation and Practical Aspects of Resource Analysis (FOPARA)*, volume 6324 of *LNCS*, pages 131–145. Springer, 2010.
15. Glynn Winskel. Event structures. In *Advances in Petri nets: applications and relationships to other models of concurrency*, pages 325–392. Springer Verlag, 1987.
16. Nobuko Yoshida, Martin Berger, and Kohei Honda. Strong normalisation in the $\pi$-calculus. In *16th IEEE Symposium on Logic in Computer Science (LICS)*, pages 311–322, 2001.

# A Annexe: M?LL and restricted type system

## A.1 MLL with linear modalities

*Proof structures* are formally defined as follows:

- The forest is seen as an acyclic graph $(V_S, E_S)$ oriented from leaves to roots, with a total order on the ingoing arcs of each node.
- We write $v^A$ to signify that vertex $v$ has label $A$. We impose that if a vertex $v^A$ is not a leaf, then it has two children $x^B$ and $y^C$ (in this order) and either $A = B \otimes C$ or $A = B \parr C$; the main connective of $A$ is the *sort* of $v$.
- Axioms are pairwise disjoint pairs of leaves labelled by dual formulas.
- Cuts are disjoint pairs of roots labelled by dual formulas.
- Boxes are disjoint non-empty sets of leaves,
- $Ax \cup Box$ forms a partition of the set of leaves.
- Each box $\beta$ has a distinguished element called its *principal port*, the others are *auxiliary ports*. Each box is associated (inductively) to a proof structure $S_\beta$ with a bijection $\phi$ from the elements of $\beta$ to the conclusions of $S_\beta$.
  - For the principal port $x^A$ of $\beta$ we have $A = ?_a B$ or $A = ¿_a B$ where $B$ is the label of $\phi(x)$;
  - for each other element $y$ of $\beta$, the labels of $y$ and $\phi(y)$ are the same.

  To each box $\beta$ we also associate a location $\ell(\beta)$, with the constraint that each location is used at most once in any given structure.

The roots that are not part of a cut are called the *conclusion nodes* of $S$. The *conclusion type* of $S$ is the multiset of the labels of its conclusion nodes.

*Proof nets* are formally defined as follows:

(ax): $(\{u^A_{ax}, v^{A^\perp}_{ax}\}, \{uv\}, \emptyset, \emptyset, \{u, v\})$ is a PN.

($\parr$): If $G = (V, E, A, P, C)$ is a PN and $u^A, v^B$ are two conclusions of $G$, then $(V \uplus \{w^{A \parr B}_{\parr}\}, E, A \cup \{uw, vw\}, P \cup \{\{uw, vw\}\}, C \setminus \{u, v\} \cup \{w\})$ is a PN.

($\otimes$): If $G = (V, E, A, P, C)$ and $G' = (V', E', A', P', C')$ are disjoint PNs, $u^A$ is a conclusion of $G$ and $v^B$ is a conclusion of $G'$, then $(V \uplus V' \uplus \{w^{A \otimes B}_{\otimes}\}, E \uplus E', A \uplus A' \uplus \{uw, vw\}, P \uplus P', (C \setminus \{u\}) \uplus (C' \setminus \{v\}) \uplus \{w\})$ is a PN.

(?): If $G = (V, E, A, P, C)$ is a PN with conclusions set $C = \{u^A, v_1^{B_1}, \ldots, v_k^{B_k}\}$, then $(\{x^{?_a A}_{?_a}(G), y_1^{B_1}, \ldots, y_k^{B_k}\}, \{uv_1, \ldots, uv_k\}, \emptyset, \emptyset, \{u, v_1, \ldots, v_k\})$ is a PN.

(*co*?): idem changing $?_a A$ by $¿_a A$.

(cut): If $G = (V, E, A, P, C)$ and $G' = (V', E', A', P', C')$ are disjoint PNs, $u^A$ is a conclusion of $G$ and $v^{A^\perp}$ is a conclusion of $G'$, then $(V \uplus V', E \uplus E' \cup \{uv\}, A \uplus A', P \uplus P', (C \setminus \{u\}) \uplus (C' \setminus \{v\}))$ is a PN.

*Cut elimination* in M?LL is formally defined as follows. Let $\pi$ and $\pi'$ be a proof structure and $c$ be a partial involution on $\mathcal{L}$. We have $\pi \to^c_{ce} \pi'$ if $\pi$ contains a cut $\kappa = \{x, y\}$ either at top level or inside a box and one of the following cases occurs (note that $x$ and $y$ may be freely exchanged):

Axiom and cut rules ($A$ is a literal $\alpha$ or a M?LLformula):

$$\frac{}{1 \vdash A^o, A^\perp}\ (\text{ax}) \qquad \frac{}{1 \vdash \alpha^p, (\alpha^\perp)^q}\ (\text{at}) \qquad \frac{P \vdash \Gamma, A^o \qquad Q \vdash A^\perp, \Delta}{P \mid Q \vdash \Gamma, \Delta}\ (\text{cut})$$

Multiplicative rules:

$$\frac{P \vdash \Gamma, A^p, B^p}{P \vdash \Gamma, (A \parr B)^p}\ (\parr) \qquad \frac{P \vdash \Gamma, A^p \quad Q \vdash B^p, \Delta}{P \mid Q \vdash \Gamma, (A \otimes B)^p, \Delta}\ (\otimes)$$

Modality rules (dereliction and codereliction):

$$\frac{P \vdash \Gamma, A^o}{a^\ell.P \vdash \Gamma^\gamma, (?_a A)^p}\ (?) \qquad \frac{P \vdash \Gamma, A}{\bar{a}^\ell.P \vdash \Gamma^\gamma, (¿_a A)^p}\ (co?)$$

**Table 2.** Inference rules in M?LL$^p$

- Multiplicative step: If $x$ and $y$ have respective sorts $\otimes$ and $\parr$, then each has two premisses, call them respectively $x_1^A, x_2^B, y_1^{A^\perp}, y_2^{B^\perp}$. Then $c = \emptyset$ and $\pi'$ is obtained by removing $\kappa$ and the nodes $x$ and $y$ and adding the cuts $\{x_1, y_1\}$ and $\{x_2, y_2\}$.
- Axiom step: If $y$ is a leaf node and it is part of an axiom $\alpha = \{y, z\}$ with $x \neq z$, then $c = \emptyset$ and $\pi'$ is obtained removing $\alpha$, $\kappa$, $y$ and $z$ and rewriting any outgoing arc of $z$ into an outgoing arc of $x$.
- Modality step: If $x$ and $y$ are principal ports of two boxes $\beta, \beta'$, then $c$ permutes $\ell(\beta)$ and $\ell(\beta')$ and $\pi'$ is obtained by replacing each box with its associated proof structure, identifying the conclusions of this structure with the ports of the box.
- Commutation step: If $x$ is the auxiliary port of a box $\beta$, call $T$ the smallest subnet of $\pi$ that contains $y$. Then $c = \emptyset$ and $\pi'$ is obtained by moving $T$ and $\kappa$ inside $\beta$, replacing the auxiliary port $x$ by one auxiliary port for each conclusion of $T$.

The *annotated cut elimination* relation $\rightarrow_{ce}^{c}$ over proof structures is the reflexive transitive closure of the rules above (if $\pi \rightarrow_{ce}^{c} \pi' \rightarrow_{ce}^{d} \pi''$ then $\pi \rightarrow_{ce}^{c \cup d} \pi''$).

### A.2 M?LL$^p$, a restriction of M?LL

The restriction to anchored proof may be design with a restriction of our type system as in Table 2. The idea is to enforce in each box an orientation from auxilary ports to principal port, for axioms on modaties. We simply use a decoration on formulas when needed. The letters $p, q, \ldots$ indicate the $o$ decoration (output) or no decoration. The sequence of formulas $\Gamma^\gamma$ is a sequence of formulas decorated with letters.

## B Detailed proofs

### B.1 Runs and pairings

**Lemma 3.** *Let $P \to^c_{ex} Q$ be an execution and $d$ be a pairing of $Q$, then* $\operatorname{dom} c \cap \operatorname{dom} d = \emptyset$ *and $c \cup d$ is a pairing of $P$. If $d$ is consistent, then so is $c \cup d$.*

*Proof.* First remark that, by definition of execution, we have $\mathcal{L}(P) = \mathcal{L}(Q) \uplus \operatorname{dom} c$, besides $\operatorname{dom} d \subset \mathcal{L}(Q)$ so the domains of $c$ and $d$ are disjoint. We can thus define the involution $c' = c \cup d$, and check that it is indeed a pairing of $P$.

Let $\ell \in \operatorname{dom} c'$. If $\ell \in \operatorname{dom} c$ then $\ell$ is the location of an action involved in the execution step, so $\operatorname{subj}_P c(\ell) = \operatorname{subj}_P \ell$ and $\operatorname{pol}_P c(\ell) = -\operatorname{pol}_P \ell$ by definition of execution, and subsequently the property holds for $c'$. Otherwise $\ell$ is in the domain of $d$, then the same property holds for $d$ in $Q$ since $d$ is a pairing of $Q$, and again we get it for $c'$ in $P$. Hence $c'$ is a pairing of $P$.

Now suppose $d$ that is consistent but $c'$ is not. Write $\ell \prec^c_P m$ if there is a location $n$ such that $\ell <_P n \sim_c m$. Then there exists a cycle $\ell_0 \prec^c_P \ell_1 \prec^c_P \cdots \prec^c_P \ell_k = \ell_0$. If all the $\ell_i$ are in $Q$ then this cycle exists in $\prec^d_Q$, which cannot be since $d$ is consistent. Since $c$ annotates a reduction of $P$, all elements of $\operatorname{dom} c$ are minimal for $\leq_P$, so the cycle cannot consist only of elements of $\operatorname{dom} c$. So we may assume $\ell_0 \in \mathcal{L}(Q)$ and $\ell_1 \in \operatorname{dom} c$. This means either $\ell_0 <_P \ell_1$ or $\ell_0 <_P c(\ell_1)$, in each case this implies that some location in $\operatorname{dom} c$ is prefixed in $P$, which is impossible. Hence $c'$ is consistent.

*Proof (proposition 1).* By iterating lemma 3, from an execution $P_0 \to^{c_1}_{ex} P_1 \to^{c_2}_{ex} \cdots \to^{c_n}_{ex} P_n$ we can deduce a pairing $c = c_1 \cup \cdots \cup c_n$ of $P_0$. This pairing represents the execution above, because it contains all the choices made during this execution. Indeed we can prove that executions that yield the same pairing are equivalent. The converse is detailed in the main text.

**Lemma 4.** *Let $P \to^{c_1}_{ex} Q_1$ and $P \to^{c_2}_{ex} Q_2$ be two executions with* $\operatorname{dom} c_1 \cap \operatorname{dom} c_2 = \emptyset$. *Then there is a unique $R$ such that $Q_1 \to^{c_2}_{ex} R$ and $Q_2 \to^{c_1}_{ex} R$.*

*Proof.* By the existence of the execution step $P \to^{c_1}_{ex} Q_1$, we know that $P$ can be written $P \equiv \bar{a}^\ell.S|a^m.T|P'$ for some name $a$ and with $\operatorname{dom} c_1 = \{\ell, m\}$. Similarly, the term $P$ can be decomposed as $P \equiv \bar{b}^{\ell'}.S' \mid b^{m'}.T' \mid P''$ for some name $b$ and with $\operatorname{dom} c_2 = \{\ell', m'\}$ to jusitfy the execution step $P \to^{c_2}_{ex} Q_2$. By hypothesis the domains of $c_1$ and $c_2$ are disjoint, so $\ell$ and $m$ are distinct from $\ell'$ and $m'$. As a consequence the term $P$ can be decomposed as $P \equiv \bar{a}^\ell.S|a^m.T|\bar{b}^{\ell'}.S'|b^{m'}.T'|P'''$ and the terms $Q_1$ and $Q_2$ have execution steps with the expected annotations, with the common reduct $R = S \mid T \mid S' \mid T' \mid P'''$. Unicity of $R$ up to structural congruence is a consequence of the fact the $c_1$ and $c_2$ completely describe which subterms of $P, Q_1, Q_2$ interact and in which way.

*Proof (proposition 2).* The pairing $c$ is the disjoint union of the atomic involutions of each step, so clearly the execution sequences are permutations of each other. Write them as $P = P_0 \to^{c_1}_{ex} P_1 \cdots \to^{c_n}_{ex} P_n$ and $P = P_0 \to^{c_{\sigma(1)}}_{ex} P'_1 \cdots \to^{c_{\sigma(n)}}_{ex} P'_n$. We now prove that the final terms $P_n$ and $P'_n$ are equal up to

structural congruence. Call $d(\sigma)$ the number of pairs $(i, j)$ such that $i < j$ and $\sigma(i) > \sigma(j)$. We proceed by induction on $d(\sigma)$. If this number is 0, then $\sigma$ is the identity function and the sequences match, so obviously we have $P_n \equiv P'_n$. Otherwise, consider a minimal $i$ such that $\sigma(i) \neq i$, hence $\sigma^{-1}(i) \neq i$ and $\sigma^{-1}(i) > i$, and let $j = \sigma^{-1}(i) - 1$. The reduction sequences match in their first $i - 1$ steps, then one has a reduction labelled $c_i$ while the other has a reduction labelled $c_{\sigma(i)}$. By repeated applications of lemma 4, we can deduce that for each $k \geq i$ there is a term $Q_k$ such that $P'_k \rightarrow_{ex}^{c_i} Q_k$ and $Q_{k-1} \rightarrow_{ex}^{c_{\sigma(k)}} Q_k$ if $k > i$:

$$\begin{tikzcd}
 & P_i \arrow[r, "c_{i+1}"] \arrow[dr, "c_{\sigma(i)}"] & \cdots & & & \\
P_{i-1} \arrow[ur, "c_i"] \arrow[dr, "c_{\sigma(i)}"'] & & Q_i \arrow[r, "c_{\sigma(i+1)}"] & \cdots \arrow[r, "c_{\sigma(j-1)}"] & Q_{j-1} \arrow[r, "c_{\sigma(j)}"] & Q_j \\
 & P'_i \arrow[ur, "c_i"] \arrow[r, "c_{\sigma(i+1)}"] & \cdots \arrow[ur, "c_i"] \arrow[r, "c_{\sigma(j-1)}"] & P'_{j-1} \arrow[ur, "c_i"] \arrow[r, "c_{\sigma(j)}"] & P'_j \arrow[ur, "c_i"] \arrow[r, "c_{\sigma(j+1)}"] & P'_{j+1}
\end{tikzcd}$$

Moreover, by construction $c_{\sigma(j+1)} = c_i$, so $P'_{j+1} \equiv Q_j$, because there is at most one possible reduction for a given annotation. Hence we can deduce a pair of reduction steps $P'_{j-1} \rightarrow_{ex}^{c_{\sigma(j+1)}} Q_{j-1} \rightarrow_{ex}^{c_{\sigma(j)}} P_{j+1}$. This yields a new reduction sequence from $P_0$ to $P'_n$ that corresponds to a new permutation $\sigma'$ of the sequence $(c_i)$, and $\sigma'$ is $\sigma$ where $\sigma(j)$ and $\sigma(j+1)$ are swapped. By definition of $j$ we have $\sigma(j) > \sigma(j+1)$ so $\sigma'(j) < \sigma'(j+1)$. For any $a \notin \{j, j+1\}$ we have $\sigma(a) < \sigma(j)$ if and only if $\sigma'(a) < \sigma'(j+1)$, and the same exchanging $j$ and $j+1$, so we have $d(\sigma') = d(\sigma) - 1$, and we can conclude by induction hypothesis.

### B.2 Typing

*Proof (proposition 3).* Clearly we can deduce the general result from the case of each individual rule. Cut elimination steps for multiplicatives and axioms do not affect the nesting of boxes, which is the only part of proofs used in term assignement, so for each such step $\pi \rightarrow_h^{\emptyset} \pi'$ we have $\lfloor \pi \rfloor = \lfloor \pi' \rfloor$, hence $\lfloor \pi \rfloor \rightarrow_{ex*}^{\emptyset} \lfloor \pi' \rfloor$ by reflexivity. For an elimination step for modalities, we have $\pi \rightarrow_h^{(\ell,m)} \pi'$ where $\ell$ and $m$ are the locations of two boxes $\beta$ and $\beta'$. By definition there is a cut between their principal ports $x$ and $y$, so these ports must have dual types $?_a A$ and $¿_a A^{\perp}$. Call $\pi_1$ and $\pi_2$ the proofs associated to $\beta$ and $\beta'$, then we have $\lfloor \beta \rfloor = a^{\ell}.\lfloor \pi_1 \rfloor$ and $\lfloor \beta' \rfloor = \bar{a}^m.\lfloor \pi_2 \rfloor$. Moreover, there is a term $P$ such that $\lfloor \pi \rfloor = \lfloor \beta \rfloor \mid \lfloor \beta' \rfloor \mid P$ so we have $\lfloor \pi \rfloor \rightarrow_{ex}^{(\ell,m)} \lfloor \pi_1 \rfloor \mid \lfloor \pi_2 \rfloor \mid P$, and the latter is equal to $\lfloor \pi' \rfloor$ by definition of the cut elimination step for modalities.

*Proof (subject reduction theorem 1).* Consider an execution step $\lfloor \pi \rfloor \rightarrow_{ex}^{(\ell,m)} P$. This step involves immediate subterms $a^{\ell}.Q$ and $\bar{a}^m.R$ for some name $a$, hence $\pi$ must contain a box at top level with principal port $x^{?_a A}$ and one with principal port $y^{¿_a A^{\perp}}$, for some formula $A$. Since $\pi$ is linear, $?_a A$ and $¿_a A^{\perp}$ do not occur in the conclusion type, so they are cut. Since $\pi$ is linear, no other boxes introducing these modalities can be in $\pi$. So $x$ and $y$ are elimination boxes for each other,

and there is a path (as of definition 11) $\rho$ from $x$ to $y$ in $\pi$. Remark that ends of $\rho$ are derelicition and codereliction on $?_a A$ and $¿_a A^\perp$ whereas all axioms along $\rho$ contain these modalities in their types. Let $\pi'$ be the rewriting of $\pi$ where such modalities are removed by rewriting axioms on $?_a A/¿_a A^\perp$ in axioms on $A/A^\perp$, and by rewriting the end boxes by their contents. Clearly $\pi'$ is a linear proof that infers $P$.

*Proof (strong subject reduction theorem 2).* Consider an execution step $\lfloor\pi\rfloor \rightarrow_{ex}^{(\ell,m)} P$. This step involves immediate subterms $a^\ell.Q$ and $\bar{a}^m.R$ for some name $a$, hence $\pi$ must contain a box at top level with principal port $x^{?_a A}$ and one with principal port $y^{¿_a A^\perp}$, for some formula $A$. Since $\pi$ is linear, $?_a A$ and $¿_a A^\perp$ do not occur in the conclusion type, so they are cut. Since $\pi$ is linear, no other boxes introducing these modalities can be in $\pi$. So $x$ and $y$ are elimination boxes for each other, and there is a path (as of definition 11) $\rho$ from $x$ to $y$ in $\pi$. For simplification we consider that boxes are replaced by their associated proof net keeping the information of auxilary and principal ports (this is more like sequent calculus derivations).

Since $x$ is at top level and cut, suppose that along $\rho$ we get through a box $\beta$ from $x$. By duality it is only with axioms on modality formulas $?_a A/¿_a A^\perp$. Moreover by typing rules going inside $\beta$ can only be done through an auxilary port. Since $\pi$ is regular, $\rho$ go out from $\beta$ through its principal port. By typing rules it is not possible to reach $y$ without enconter before a principal port of a box eliminating $\beta$. Moreover this is only possible using an axiom on modality formulas $?_a A/¿_a A^\perp$ in this box. Remak that since $\pi$ is regular and $\beta$ has axioms on modalities, corresponding elimination box is simple. Then to use an axiom on modalities contradict that $\pi$ is regular. Then there is no box traversal along $\rho$. Then by typing $\rho$ is a multiplicative cut path whose cut elimination $\rightarrow_h^\emptyset$ until $x$ and $y$ preserves $\lfloor\pi\rfloor$ as well as regularity and linearity of proof.