Polarized Proof Nets with Cycles and Fixpoints Semantics

Raphaël Montelatici

Équipe Preuves, Programmes et Systèmes*
Raphael.Montelatici@pps.jussieu.fr

Abstract. Starting from Laurent's work on *Polarized Linear Logic*, we define a new polarized linear deduction system which handles recursion. This is achieved by extending the *cut*-rule, in such a way that iteration unrolling is achieved by cut-elimination. The *proof nets* counterpart of this extension is obtained by allowing oriented cycles, which had no meaning in usual polarized linear logic. We also free proof nets from additional constraints, leading up to a correctness criterion as straightforward as possible (since almost all proof structures are correct). Our system has a sound semantics expressed in traced models.

1 Introduction

This paper investigates two different topics, proof nets and fixpoints semantics, in order to present a new syntax, *polarized proof nets with cycles*, as a way of encoding recursion within a logical-like framework.

Proof nets. Girard's Linear Logic [4] introduced new syntactic objects, proof nets, that represent proofs by means of graphs. They provide a much less sequential syntax than sequent calculus and yield a more convenient way to study cut-elimination (see for instance [3]). However, LL as a whole is pretty complex. A new system named Polarized Linear Logic (LLP), introduced by Laurent, arose from the study of a sub-system of LL ruled by polarity constraints. LLP enjoys simpler properties than LL, and is expressive enough to interpret classical systems such as Girard's LC or Parigot's $\lambda\mu$-calculus [14].
Laurent has given a fairly comprehensive description of LLP proof nets [9], which enjoy a good accommodation of additive connectives, a confluent and strongly normalizing procedure of cut-elimination and a simple correctness criterion.
Fixpoints semantics. Domain theory generally interprets recursive programs by means of least fixed-points of continuous functions. In categorical semantics, fixpoints are axiomatized as *fixpoint operators*. Simpson and Plotkin described in [16] various classes of fixpoint operators and provided axiomatic methods to construct them. Alternatively, *trace operators* [8] interpret recursive functions as well as cycling data structures [5].

* CNRS & Université Paris VII, Case 7014, 2 place Jussieu, 75251 Paris cedex 05.

The proof nets with cycles introduced in this paper are a new instance of such cyclic structures. As such, they have a natural interpretation in traced cartesian closed categories. The idea of adding a fixpoint primitive to LLP and the way of doing it is a real semantic backlash. In [13] we studied LLP semantics in *categories of continuations*, following Laurent's work on polarized games [10] as well as Melliès and Selinger's ideas [12]. We noticed that relaxing the *totality* constraint on strategies corresponds to adding strategies computed by fixpoints operations. The following step was to move from semantics to syntax, add a fixpoint deduction rule to a polarized logical system inspired by LLP, then study the associated proof nets and correctness criterion. Polarized proof nets with cycles appear as an elegant solution. The idea that cyclic nets could handle recursion has been suggested informally by Danos. Today the modern theory of polarized proof nets allows to formalize and study them.

2 Polarized Linear Logic

Here we briefly recall features of LLP that are relevant to our presentation. The reader should consult [9] for a complete account. In this paper, we only deal with MELLP, the *Multiplicative Exponential Polarized Linear Logic* fragment of LLP, though most of the properties stated below still hold for LLP.

2.1 MELLP syntax

The formulas of MELLP are divided into positive and negative ones:

$$\begin{array}{ccccccc} P & ::= & 1 & \mid & P \otimes P & \mid & !N \\ N & ::= & \bot & \mid & N \parr N & \mid & ?P \end{array}$$

Notational conventions. We adopt the following conventions: P, Q denote positive formulas and N, M denote negative formulas. Γ and Δ stand for multisets of formulas (of any polarity) and $\mathcal{N}$ for a multiset of negative only formulas. The perpendicular symbol $\perp$ is used as a meta-operation negating formulas, according to the usual De Morgan rules of LL.

The deduction rules presented in figure 1 allow to prove sequents of shape $\vdash N_1, \ldots, N_k$ or $\vdash N_1, \ldots, N_k, P$ (*i.e.* with at most one positive formula).

2.2 Polarized proof nets

MELLP proof structures are oriented graphs inspired by LL proof structures. LLP introduces an orientation on edges which is determined by formulas polarities. Check figure 2 for the grammar of MELLP nodes.
As for LL proof structures, !-nodes are associated to !-boxes, such that edges crossing auxiliary doors must be labelled by negative formulas, see figure 3. The *level* of a node is defined to be the number of nested boxes which contain it.

$$\frac{}{\vdash N, N^{\perp}}ax \qquad \frac{\vdash \Gamma, N \quad \vdash \Delta, N^{\perp}}{\vdash \Gamma, \Delta}cut$$

$$\frac{\vdash \Gamma, P \quad \vdash \Delta, Q}{\vdash \Gamma, \Delta, P \otimes Q}\otimes \qquad \frac{\vdash \Gamma, N, M}{\vdash \Gamma, N \parr M}\parr$$

$$\frac{}{\vdash 1}1 \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp}\perp$$

$$\frac{\vdash \mathcal{N}, N}{\vdash \mathcal{N}, !N}! \quad \frac{\vdash \Gamma, P}{\vdash \Gamma, ?P}? \quad \frac{\vdash \Gamma, N, N}{\vdash \Gamma, N}c \quad \frac{\vdash \Gamma}{\vdash \Gamma, N}w$$

Fig. 1. MELLP deduction rules

The obvious transformation mapping MELLP deduction trees to proof structures is not onto and thus defines a strict subset of proof structures: *proof nets.* A so-called *correctness criterion* decides which proof structures are proof nets.

Definition 2.1 (Correctness graph). *The correctness graph of a proof structure is obtained by replacing* !*-boxes of the* 0*-level by generalized axiom nodes.*

Definition 2.2 (Correctness criterion). *Let* $C(\mathcal{R})$ *be the following property: The correctness graph of the proof structure* $\mathcal{R}$ *has no oriented cycle and has exactly one initial node*[1] *which is not a* w*-node or a* $\perp$*-node.*
A proof structure $\mathcal{R}$ *is a proof net if and only if it satisfies* $C(\mathcal{R})$ *and all the proof structures enclosed in any* !*-box inside* $\mathcal{R}$ *do so.*

This criterion is really simple, because it does not require to switch proof nets, like LL criterion does.

We recall the important notion of *positive trees*:

Definition 2.3 (Positive trees). *Positive trees are proof structures with one positive conclusion, inductively defined as:*

- *ax nodes,* 1*-nodes and* !*-boxes wrapping a proof structure are positive trees,*
- *two positive trees on top of a* $\otimes$*-node form a positive tree.*

Any positive edge within a proof structure is the root of a positive tree. Positive trees are well-delimited structures that allow copying and deleting: from this point of view, they generalize !-boxes. See figure 3 for their representation.

A set of *cut-elimination rules* makes proof nets a rewriting system, which task is to eliminate the *cut*-nodes. In the case of LLP and MELLP, cut-elimination is confluent and strongly normalizing. In figure 4, we give the rules for MELLP (note that polarized systems take advantage of polarities and use positive trees to express the cut-elimination rules).

[1] with respect to the orientation relation on nodes

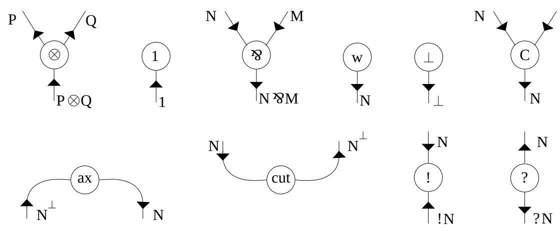

Fig. 2. Nodes

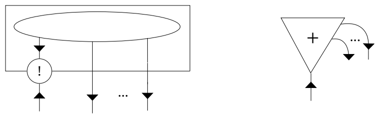

Fig. 3. Exponential box and positive tree representations

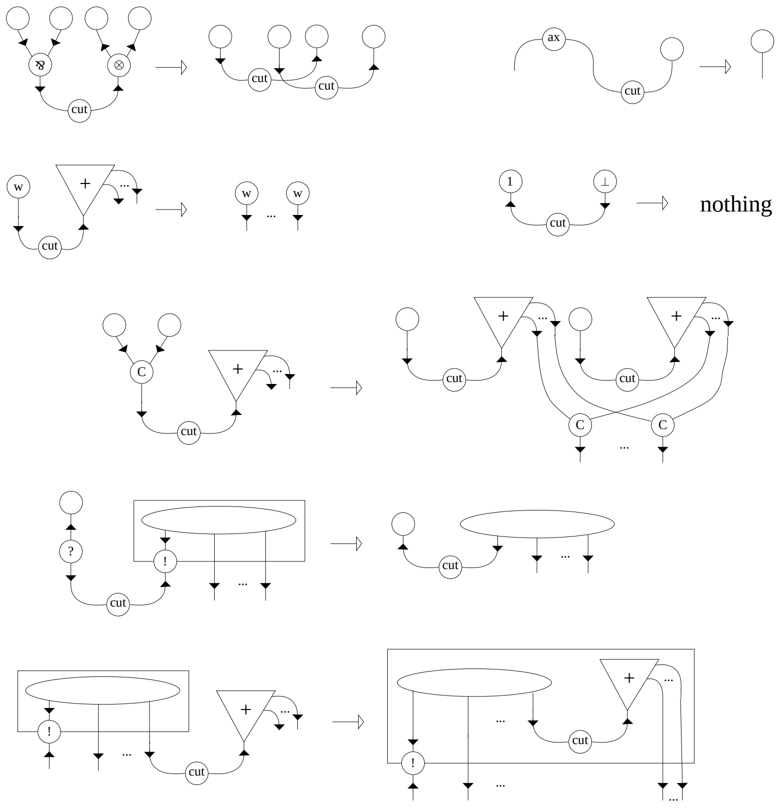

Fig. 4. Cut-elimination rules

3 The Semantic Pathway from LLP to MELLP with Cycles

This paper is about adding fixpoints to LLP. It is well-known that fixpoints can only be computed on a category of domains, not on a category of predomains. Logically, this means moving from LLP to MELLP, that is forgetting the additive connective $\oplus$.

Modulo translation, LLP and the extended $\lambda\mu$-calculus of Selinger are equivalent. Hence LLP can be interpreted in any model of the $\lambda\mu$-calculus, like *control categories* [15]. Selinger proves a representation theorem, stating that every control category is isomorphic (in the sense of control categories) to some category $R^{\mathcal{C}}$ where $\mathcal{C}$ is a *response category*. Categorically, moving from LLP to MELLP means relaxing the notion of response categories, by discarding coproducts as well as the so-called mono requirement [15]. This leads to *categories of continuations*, in the sense of Hofmann and Streicher [7]:

Definition 3.1. *A category of continuations $\mathcal{C}$ is a category:*

- *with a cartesian product $\otimes$ and a terminal object 1,*
- *with a distinguished object R and a counter-variant functor $R^{(\_)} : \mathcal{C}^{op} \to \mathcal{C}$ such that there is a natural family of bijections $\Lambda_{A,B}$*

$$\frac{A \otimes B \to R}{A \to R^B}$$

Positive formulas of MELLP are interpreted according to the following rules:

$$[\![1]\!] = 1 \qquad [\![P \otimes Q]\!] = [\![P]\!] \otimes [\![Q]\!] \qquad [\![!N]\!] = R^{[\![N^\perp]\!]}$$

A deduction tree of conclusion sequent $\vdash N_1, \ldots, N_k, P$ is interpreted by a morphism $[\![N_1^\perp]\!] \otimes \ldots \otimes [\![N_k^\perp]\!] \to [\![P]\!]$ and a deduction tree of conclusion sequent $\vdash N_1, \ldots, N_k$ is interpreted by a morphism $[\![N_1^\perp]\!] \otimes \ldots \otimes [\![N_k^\perp]\!] \to R$.

Important examples of categories of continuations are cartesian closed categories. Let us recall the definition of a fixpoint operator in a cartesian category:

Definition 3.2 (Fixpoint operator). *Let $(\mathcal{D}, \times, 1)$ be a cartesian category with a terminal object. A parameterized fixpoint operator is a family of functions $(\_)^\nabla : \mathcal{D}(X \times A, A) \to \mathcal{D}(X, A)$ such that:*

1. *(Fixpoint property) For all $f : X \times A \to A$, $< id_X, f^\nabla >; f = f^\nabla$*
2. *(Naturality) For all $f : X \times A \to A$ and $g : Y \to X$, $(g \times A; f)^\nabla = g; f^\nabla$*

An interesting example: we noticed in [13] that the category of continuations associated to Hyland-Ong games and innocent strategies [10] coincides with the underlying control category; and thus admits the fixpoint operator associated to Y in the usual interpretation of PCF.
Adding a trace to the semantics may be understood from several perspectives.

Proof-theoretically, it means adding the following fixpoint deduction rule to MELLP sequent calculus:

$$\frac{\vdash \Gamma, P^{\perp}, P}{\vdash \Gamma, P} Y$$

Syntactically, it means extending polarized proof nets with so-called Y-boxes; and adding the reduction rule of figure 5 to usual proof net cut-elimination [13].

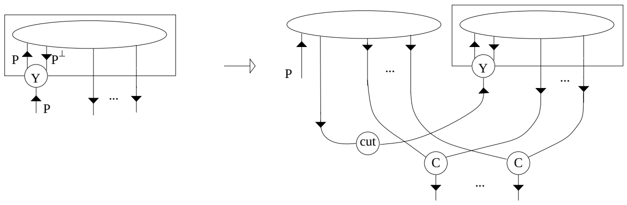

Fig. 5. Y-box and associated reduction rule

Here, we want to go one step further and set up a system that simulates the dynamical behavior of the Y-box by means of cut-elimination rules only. As duplication is typically performed by the elimination of a *cut*-node below a contraction node, it looks like the proof net of figure 6 is a nice candidate.

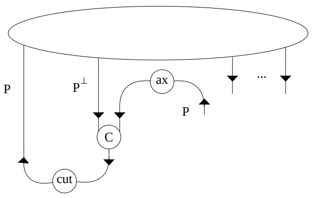

Fig. 6. An alternative for Y

This MELLP proof structure is generally not a MELLP proof net because it may contain oriented cycles. In order to see such cyclic structures as “proofs”, we shall introduce a deduction system with (1) a *mix*-rule which allows putting several proofs side by side and (2) a relaxed *cut*-rule which applies to a single premise.

4 Multiplicative Exponential Polarized Linear Logic with Cycles

Our system is based on MELLP and is called MELLPC (MELLP *with cycles*). Since cut-elimination is not strongly normalizing in the presence of recursion, MELLPC is closer to a programming language, but still enjoys a flavor of logic.

4.1 Sequents-style syntax

We keep MELLP class of formulas:

$$\begin{array}{lcl} P & ::= & 1 \quad | \quad !N \quad | \quad P \otimes P \\ N & ::= & \bot \quad | \quad ?P \quad | \quad N \parr N \end{array}$$

The sequents are of the form $\{n\} \vdash \Gamma$ where Γ is a multiset of polarized formulas and n is a natural number. The deduction rules are presented in figure 7.

$$\frac{}{\{0\} \vdash \Gamma, \Gamma^{\perp}}ax \qquad \frac{\{n\} \vdash \Gamma \quad \{m\} \vdash \Gamma'}{\{n+m\} \vdash \Gamma, \Gamma'}mix \qquad \frac{\{n\} \vdash \Gamma, N, N^{\perp}}{\{n\} \vdash \Gamma}cut$$

$$\frac{\{n\} \vdash \Gamma, P, Q}{\{n\} \vdash \Gamma, P \otimes Q}\otimes \qquad \frac{\{n\} \vdash \Gamma, N, M}{\{n\} \vdash \Gamma, N \parr M}\parr$$

$$\frac{}{\{0\} \vdash 1}1 \qquad \frac{\{n\} \vdash \Gamma}{\{n\} \vdash \Gamma, \bot}\bot$$

$$\frac{\{1\} \vdash \mathcal{N}, N}{\{0\} \vdash \mathcal{N}, !N}! \qquad \frac{\{n\} \vdash \Gamma, P}{\{n+1\} \vdash \Gamma, ?P}? \qquad \frac{\{n\} \vdash \Gamma, N, N}{\{n\} \vdash \Gamma, N}c \qquad \frac{\{n\} \vdash \Gamma}{\{n\} \vdash \Gamma, N}w$$

Fig. 7. MELLPC deduction rules

Remark 4.1. The natural numbers labelling the sequents hold a counter of the number of ?-rules used in the sub-tree above each sequent, which are not above any !-rule in the same sub-tree. We could have discarded these numbers since they are determined by the deduction rules, but we keep them anyway, because they provide a convenient way to express the new !-rule and also because they directly take part in the semantics of proofs (see next subsection).

Remark 4.2. Since multisets can be empty, the axiom rule may be instantiated in order to prove the empty sequent $\{0\} \vdash$ in one step. Even if we disallow such a use of the axiom rule, our system would be able to prove the empty sequent anyway, by applying the *cut*-rule to the sequent $\{0\} \vdash N, N^{\perp}$ obtained after an axiom rule on the formula N. From a programming point of view, these proofs correspond to non-terminating recursions, *i.e.* diverging fixpoint applications.

Proposition 4.3. *By labelling* MELLP *sequents* ($\vdash N_1, \ldots, N_k$ *becoming* $\{1\} \vdash N_1, \ldots, N_k$ *and* $\vdash N_1, \ldots, N_k, P$ *becoming* $\{0\} \vdash N_1, \ldots, N_k, P$*),* MELLP *can be embedded in* MELLPC*, making every* MELLP *deduction rule admissible.*

Proof. By induction. For instance, the usual *cut*-rule is simulated by:

$$\dfrac{\dfrac{\{0\} \vdash \Gamma, N \qquad \{i\} \vdash \Delta, N^{\perp}}{\{i\} \vdash \Gamma, \Delta, N, N^{\perp}}\,mix}{\{i\} \vdash \Gamma, \Delta}\,cut \qquad \text{where } i = 0 \text{ or } 1$$

4.2 Semantics in traced cartesian closed categories

We designed MELLPC in the light of both semantics and proof net technology. On one hand we want a correctness criterion as simple as possible, and on the other hand, we want to stick to a straightforward semantics. In order to allow more correct proof structures, the MELLPC dereliction ?-rule handles a context $\Gamma^{\perp}$ of negative formulas and Δ of positive formulas. Semantically, this implies to interpret:

$$\dfrac{\Gamma \rightarrow \Delta \otimes A}{\Gamma \otimes R^{A} \rightarrow \Delta \otimes R}$$

This can be achieved in cartesian closed categories but not in categories of continuations.

We recall the definition of *traced symmetric monoidal categories*, as found in [5]. This definition is a slight adaptation from the original one given in [8].

Definition 4.4. *A symmetric monoidal category* $(\mathcal{C}, \otimes, 1)$ *is said to be* traced *if it is equipped with a natural family of functions, called a* trace*,*

$$Tr^{A}_{\Gamma,\Delta} : \mathcal{C}(\Gamma \otimes A, \Delta \otimes A) \longrightarrow \mathcal{C}(\Gamma, \Delta)$$

which satisfies the following properties:

1. ***Vanishing**:* $Tr^{1}_{A,B}(f) = f$*, for all* $f : A \longrightarrow B$
   and $Tr^{X \otimes Y}_{A,B}(f) = Tr^{X}_{A,B}(Tr^{Y}_{A \otimes X, B \otimes X}(f))$ *for all* $f : A \otimes X \otimes Y \longrightarrow B \otimes X \otimes Y$
2. ***Superposing**:* $Tr^{X}_{C \otimes A, C \otimes B}(id_c \otimes f) = id_C \otimes Tr^{X}_{A,B}(f)$
   for all $f : A \otimes X \longrightarrow B \otimes X$
3. ***Yanking**:* $Tr^{X}_{X,X}(s_{X,X}) = id_X$
   (where $s_{X,X} : X \otimes X \longrightarrow X \otimes X$ *is the symmetry morphism)*

Note that the naturality of $Tr^{A}_{\Gamma,\Delta}$ in Γ, Δ and A is usually called *left tightening*, *right tightening* and *sliding*, respectively.

On the semantical side, trace operators perfectly match the extended *cut*-rule of MELLPC. It is worth noting that in the case of cartesian closed categories, trace operators are equivalent to *Conway operators*, which are fixpoints operators with some additional properties [5].

We interpret MELLPC in *traced cartesian closed categories* with a distinguished object R, as follows:

• Positive formulas are interpreted according to the following rules:

$$[\![1]\!] = 1 \qquad [\![P \otimes Q]\!] = [\![P]\!] \otimes [\![Q]\!] \qquad [\![!N]\!] = [\![N^\perp]\!] \Rightarrow R$$

• A deduction tree of root $\{n\} \vdash N_1, \ldots, N_k, P_1, \ldots, P_l$ is interpreted by a morphism:

$$[\![N_1^\perp]\!] \otimes \ldots \otimes [\![N_k^\perp]\!] \longrightarrow [\![P_1]\!] \otimes \ldots \otimes [\![P_l]\!] \otimes R \otimes \ldots \otimes R$$

where the number of R's in the right-hand side product is equal to n. The semantics of an empty product is equal to the terminal object 1, as usual.

The interpretation morphism is constructed by induction on the deduction tree according to the rules of figure 8. Although these rules build interpretation morphisms that depend on an assumed ordering of the multisets composing the sequents, the axiomatic properties of trace operators allow to interpret the sequents up to commutations of formulas.

Rule	Premise morphism(s)	Conclusion morphism
ax		$\Gamma \xrightarrow{id_\Gamma} \Gamma$
1		$1 \xrightarrow{id_1} 1$
mix	$\Gamma \xrightarrow{f} \Delta$ and $\Gamma' \xrightarrow{g} \Delta'$	$\Gamma \otimes \Gamma' \xrightarrow{f \otimes g} \Delta \otimes \Delta'$
cut	$\Gamma \otimes A \xrightarrow{f} \Delta \otimes A$	$\Gamma \xrightarrow{Tr(f)} \Delta$
c	$\Gamma \otimes A \otimes A \xrightarrow{f} \Delta$	$\Gamma \otimes A \xrightarrow{(id_\Gamma \otimes d_A);f} \Delta$
w	$\Gamma \xrightarrow{f} \Delta$	$\Gamma \otimes A \xrightarrow{(id_\Gamma \otimes e_A);f} \Delta$
!	$\Gamma \otimes A \xrightarrow{f} R$	$\Gamma \xrightarrow{\Lambda(f)} A \Rightarrow R$
?	$\Gamma \xrightarrow{f} \Delta \otimes A$	$\Gamma \otimes (A \Rightarrow R) \xrightarrow{\Lambda^{-1}(f;\Delta \otimes \nu_A;m_{A,\Delta})} \Delta \otimes R$
$\otimes$, $\invamp$, $\perp$	nothing to do	

where $A \xrightarrow{d_A} A \otimes A$ and $A \xrightarrow{e_A} 1$ are respectively the diagonal and the erasing morphism, Λ denotes curryfication, $\nu_A : A \longrightarrow (A \Rightarrow R) \Rightarrow R$ is a canonical morphism and $m_{A,\Delta}$ is computed as follows (here we need cartesian closeness):

$$\cfrac{\cfrac{\cfrac{(A \Rightarrow R) \Rightarrow R \longrightarrow (A \Rightarrow R) \Rightarrow R}{((A \Rightarrow R) \Rightarrow R) \otimes (A \Rightarrow R) \longrightarrow R}}{\Delta \otimes ((A \Rightarrow R) \Rightarrow R) \otimes (A \Rightarrow R) \longrightarrow \Delta \otimes R}}{m_{A,\Delta} : \Delta \otimes ((A \Rightarrow R) \Rightarrow R) \longrightarrow (A \Rightarrow R) \Rightarrow (\Delta \otimes R)}$$

Fig. 8. Interpretation of rules in traced cartesian closed categories

5 Polarized Proof Nets with Cycles

5.1 Definition

Definition 5.1. *The proof structures of* MELLPC *are exactly those of* MELLP*, except that we also allow the empty proof structure.*

Definition 5.2 (Proof nets). *We define* $\mathfrak{N}$ *to be the straightforward transformation which maps* MELLPC *deduction trees to proof structures. The set of proof nets is the image of* $\mathfrak{N}$*.*

Proposition 5.3. *The translation* $\mathfrak{N}$ *maps a* MELLPC *deduction tree of conclusion* $\{n\} \vdash \Gamma$ *to a proof net of conclusions* Γ*.*

The conclusions of the proof net $\mathfrak{N}(\pi)$ associated to a proof π do not keep track of the natural numbers labelling the sequents. However the label of the conclusion sequent of π is equal to $r(\mathfrak{N}(\pi))$, where r is defined as:

Definition 5.4. *For any proof structure* $\mathcal{R}$*, let* $r(\mathcal{R})$ *be the number of* ?*-nodes lying at the level* 0 *of* $\mathcal{R}$*.*

5.2 Correctness criterion

Proposition 5.5. *Each connected component of a proof structure contains at most one node that falls in either of the following categories:*

- *nodes having a positive conclusion that is a conclusion of the proof structure,*
- ?*-nodes of the* 0*-level.*

This has to be compared with MELLP's correctness criterion (definition 2.2): initial nodes in the correctness graph fall in either of the above categories. Allowing juxtaposition of proof nets by means of the *mix*-rule discards the requirement of having only one such node. The acyclicity constraint is also discarded by our system. This yields a much simpler correctness criterion:

Definition 5.6 (Correctness criterion). *A proof structure* $\mathcal{R}$ *satisfies the correctness criterion (i.e. is* correct*) when:*
all proof structures $\mathcal{R}_0$ *enclosed in a* !*-box of any level of* $\mathcal{R}$ *satisfy* $r(\mathcal{R}_0) = 1$*.*

The criterion merely makes sure that proof net boxing is performed consistently with the restriction enforced on the !-rule of MELLPC.

Theorem 5.7 (Proof nets characterization). *A proof structure is a proof net if and only if it satisfies the correctness criterion.*

It is easy to check that proof nets satisfy the correctness criterion. The converse is a consequence of the following sequentialization property.

Proposition 5.8 (Sequentialization). *From any correct proof structure $\mathcal{R}$, we can compute a proof π of* MELLPC *such that $\mathfrak{N}(\pi) = \mathcal{R}$, by means of the sequentialization algorithm described below. If $\mathcal{R}$ has conclusions $N_1, \ldots, N_k, P_1, \ldots, P_l$ then π has conclusion $\{r(\mathcal{R})\} \vdash N_1, \ldots, N_k, P_1, \ldots, P_l$.*

Proof. The sequentialization algorithm S builds the deduction tree π from root up to leaves. It picks a node out of the proof structure $\mathcal{R}$ that corresponds to the final deduction rule of π and goes on building the deduction tree by a recursive call on the remaining sub-structure(s). More precisely, if $\mathcal{R}$ has a terminal node[2] N which is different from an ax-node, 1-node or !-node, then S picks out one of these nodes. If $\mathcal{R}$ does not have such terminal nodes, then it is a mere juxtaposition of ax-nodes, 1-nodes and !-boxes. Then S is recursively called on each sub-structure enclosed in the !-boxes, and computes the desired deduction tree using mix-rules. At each step, the deduction rule added to π is guaranteed to be consistently applied because the decomposition performed by S preserves correctness.

Remark 5.9. This algorithm is not deterministic. Anyway for any output $S(\mathcal{R})$ produced by a particular run of S on input $\mathcal{R}$, we have $\mathfrak{N}(S(\mathcal{R})) = \mathcal{R}$.

5.3 Proof nets semantics

The proof nets syntax is closer to semantics than the sequent calculus syntax:

Theorem 5.10. *Let π and π' be two proofs of* MELLPC*. We have*

$$\mathfrak{N}(\pi) = \mathfrak{N}(\pi') \Longrightarrow [\![\pi]\!] = [\![\pi']\!]$$

This result allows to define the interpretation of polarized proof nets with cycles in traced cartesian closed categories:

Definition 5.11. *Let $\mathcal{R}$ be a polarized proof net with cycles. We define its interpretation $[\![\mathcal{R}]\!]$ to be any $[\![\pi]\!]$ where π is a proof of* MELLPC *such that $\mathfrak{N}(\pi) = \mathcal{R}$ (the sequentialization proposition ensures that such a π always exists).*

6 Cut Elimination

We now describe the dynamic behavior of MELLPC by explaining the cut-elimination process. We keep the standard MELLP cut-elimination rules of figure 4 unchanged. One need to be careful however. The MELLP rewriting rules involve premise nodes belonging to the contextual net and conclusion edges that may be linked to nodes of the contextual net. When dealing with cycles, some of those premise nodes and conclusion nodes might be equal. For instance, keeping the contraction rule unchanged enables to simulate Y by the MELLPC proof net of figure 6. The cut-elimination patterns of figure 4 lead to corresponding reduction rules for MELLPC in all cases but two pathological cases: the so-called axiom loop and exponential loop. We give two solutions to reduce these patterns.

[2] Here by terminal node, we mean a node which does not have any conclusion (such as the *cut*-node), or a node whose conclusions are all conclusions of the proof structure.

6.1 MELLPC with β_0-reduction

The axiom and exponential loops can be replaced by unambiguous patterns by expansing an edge into a sequence of an axiom node and a cut node. This trick yields the rewriting relation β_0, which handles axiom and exponential loops as displayed on figure 9. As a special case of the exponential loop reduction, a !-box whose !-node is cut against one of its own auxiliary doors is reduced to itself.

Proposition 6.1. *(MELLPC,β_0) satisfies the following properties: (1) it preserves proof nets correctness, (2) it is sound with respect to traced cartesian closed categories, (3) it is not locally confluent.*

Proof. Preservation of proof correctness is easy to check, thanks to the simplicity of the correctness criterion. The soundness result follows from the axiomatic properties of trace operators. Figure 10 shows a counterexample of local confluence, where *tt* and *ff* are two distinct normal nets of the same negative type.

6.2 MELLPC with β-reduction

The axiom and exponential loops are some kinds of "ghost nets" which only possible interaction is to swallow through their auxiliary doors some positive trees. The net enclosed in the !-box cannot be probed by its context (this is confirmed by both syntax and semantics). This observation leads up to the cut-elimination rules β which policy is to delete ghost nets, see figure 11.

Proposition 6.2. *(MELLPC,β) satisfies the following properties: (1) it preserves proof nets correctness, (2) it is sound with respect to traced cartesian closed categories, (3) it is locally confluent, (4) it is not confluent.*

Proof. The local confluence follows from an overview of critical pairs: each diagram can be closed by means of a few rewriting steps. See figure 12 for a counterexample of confluence, where *tt*+ and *ff*+ are two distinct normal proof nets of the same positive type. This should be compared with the counterexample in cyclic λ-calculus observed by Ariola and Klop in [1].

6.3 A confluent subsystem: MELLPY

In order to achieve confluence, we may define Y-cycles as oriented cycles that step over exactly one contraction node and one !-box on their way.

Definition 6.3. *Let MELLPY be the subset of MELLPC proof nets such that every cycle is a Y-cycle.*

The axiom and exponential loops are not part of MELLPY; and MELLPY is closed under cut-elimination. Thus β_0 and β agree on MELLPY.

Proposition 6.4. *(MELLPY,β) satisfies the following properties: (1) it preserves proof nets correctness, (2) it is sound with respect to traced cartesian closed categories, (3) it is confluent.*

Proof. The confluence follows from a direct and tedious proof, based on the decreasing diagram method and a sharp analysis of commuting reduction steps.

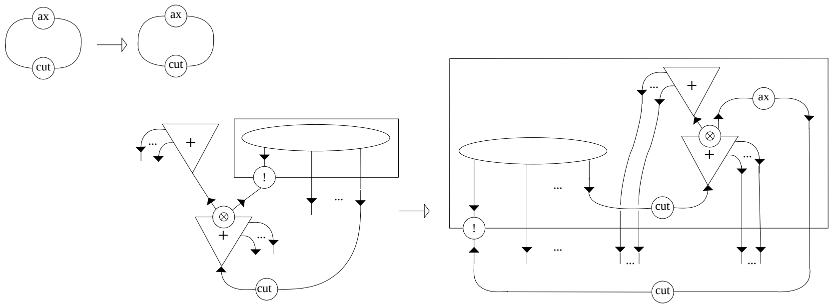

Fig. 9. β_0-reduction on axiom and exponential loops

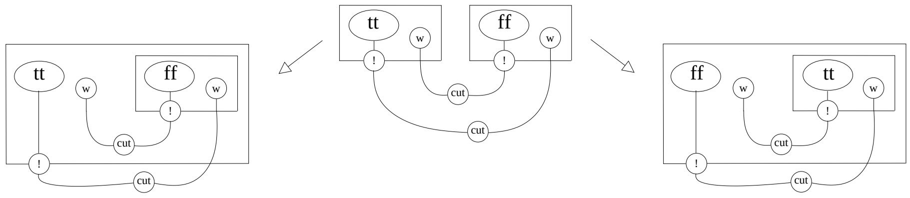

Fig. 10. Counterexample: β_0 is not locally confluent

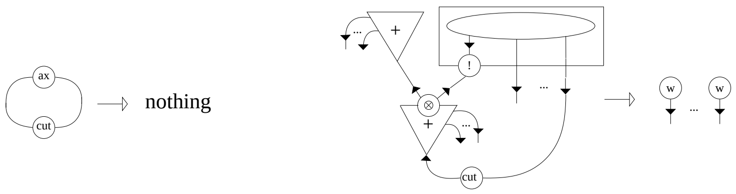

Fig. 11. β-reduction on axiom and exponential loops

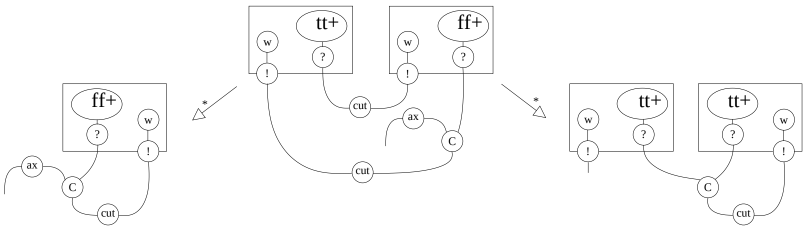

Fig. 12. Counterexample: β is not confluent

7 Interpretation of $\lambda\mu$-calculus with Fixpoint

Laurent showed [11, 9] that MELLP is a suitable target for interpreting both call-by-name and call-by-value $\lambda\mu$-calculus.

The call-by-name interpretation $(.)^-$ uses negative formulas for interpreting types: $(A \to B)^- = ?(A^-)^\perp ⅋ B^-$ and translates a typing derivation of conclusion $\Gamma \vdash t : A \mid \Delta$ into a proof net of conclusions $?(\Gamma^-)^\perp$, Δ^- and A^-.
On the other hand, the call-by-value interpretation $(.)^+$ uses positive formulas for interpreting types: $(A \to B)^+ = !((A^+)^\perp ⅋ ?(B^+))$ and translates a typing derivation of conclusion $\Gamma \vdash t : A \mid \Delta$ into a proof net of conclusions $(\Gamma^+)^\perp$, $?(\Delta^+)$ and $?(A^+)$.

It is possible to extend the $\lambda\mu$-calculus with fixpoint terms Y_A of type $(A \to A) \to A$ and interpret the resulting call-by-name and call-by-value calculi in MELLPC, using the call-by-name and call-by-value variants of figure 6, see figure 13. It is easy to check that these interpretations actually target MELLPY. These translations do not yield exact simulation properties, but the following equational properties do hold:

Proposition 7.1. *In the call-by-value setting:* $(Y(\lambda x.t))^+ =_\beta ((\lambda x.t)(Y(\lambda x.t)))^+$.

Proposition 7.2. *In the call-by-name setting:* $(Yf)^- \simeq (f(Yf))^-$.

Where $\simeq$ is $=_\beta$ together with additional structural rules, generated by a η-rule and a generalized auxiliary exponential cut elimination rule, which are sound in traced cartesian closed categories (details omitted here).

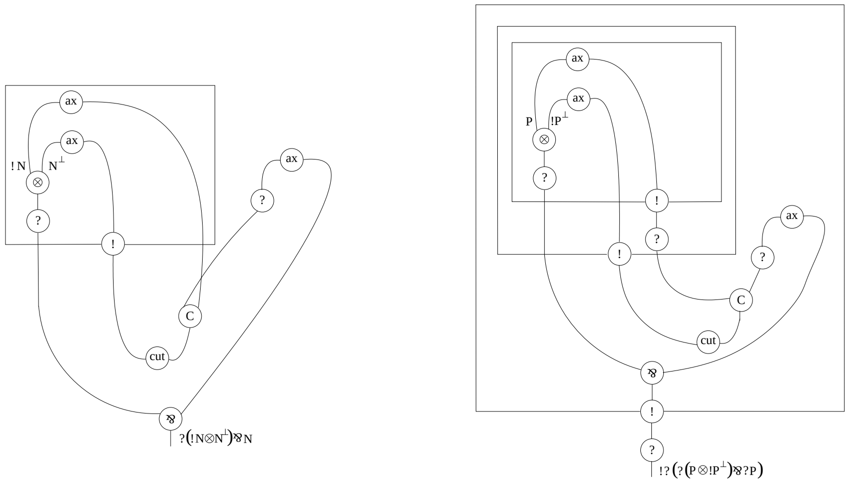

Fig. 13. The Y proof nets: call-by-name (left) and call-by-value (right)

8 Conclusion

We extend polarized linear logic with a fixpoint operator, and obtain a deduction system with (1) a sound denotational semantics in traced cartesian closed categories, (2) a theory of cyclic proof nets, (3) a correctness criterion in the spirit of Laurent's criterion. One main observation is that one needs to move from response categories to traced cartesian closed categories in order to reach that nice balance (see subsection 4.2). Polarized linear logic provides a logical account of the (non-commutative) continuation monad. This should be related to alternative axiomatics of commutative and non-commutative monads by Benton and Hyland [2], and also by Hasegawa and Kakutani [6]. It should be informative to translate various existing calculi with recursion into proof nets with cycles, and study the resulting equational properties.

References

1. Z. M. Ariola and J.W. Klop, Cyclic Lambda Graph Rewriting. In *Proc. of the Eight IEEE Symposium on Logic in Computer Science*, Paris, July 1994.
2. N. Benton and M. Hyland, Traced Premonoidal Categories (Extended Abstract). Workshop on Fixed Points in Computer Science (FICS 2002). July 2002.
3. V. Danos, Une Application de la Logique Linéaire à l'Étude des Processus de Normalisation (principalement du λ-calcul). Thèse de Doctorat, Université Paris 7, 1990
4. J-Y. Girard, Linear Logic. *Theoretical Computer Science* 50 :1-102, 1987.
5. M. Hasegawa, Recursion from Cyclic Sharing: Traced Monoidal Categories and Models of Cyclic Lambda Calculi. In *Proc. 3rd International Conference on Typed Lambda Calculi and Applications*, volume LNCS1210. Springer, 1997.
6. M. Hasegawa and Y. Kakutani, Axioms for recursion in call-by-value. Higher-Order and Symbolic Computation 15(2/3):235-264, September 2002
7. M. Hofmann and T. Streicher, Continuation models are universal for lambda-mu-calculus. In *Proc. LICS '97*, Warsaw, IEEE, pp. 387-397.
8. A. Joyal, R. Street and D. Verity, Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3), pages 447-468, 1996.
9. O. Laurent, *Etude de la polarisation en logique.* PHD thesis, Université Aix-Marseille II, March 2002.
10. O. Laurent, Polarized games (extended abstract). *Seventeenth annual IEEE symposium on Logic In Computer Science*, p. 265-274. IEEE Computer Society, 2002.
11. O. Laurent, Polarized proof nets and $\lambda\mu$-calculus. *Theoretical Computer Science*, 290(1):161-188, Dec. 2002.
12. P-A. Melliès and P. Selinger, Polarized categories and polarized games. Forthcoming paper, 2003.
13. R. Montelatici, Présentation axiomatique de théorèmes de complétude forte en sémantique des jeux et en logique classique. Mémoire de DEA, Univ. Paris 7, 2001.
14. M. Parigot, $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Proc. of International Conference on Logic Programming and Automated Deduction*, volume LNCS624. Springer, 1992.
15. P. Selinger, Control Categories and Duality: on the Categorical Semantics of the Lambda-Mu Calculus. *Mathematical Structures in Computer Science*, 2001.
16. A. K. Simpson and G. D. Plotkin, Complete Axioms for Categorical Fixed-Point Operators. In *Logic in Computer Science*, pages 30-41, 2000.