Proof-Nets and the Call-by-Value Lambda-Calculus

Beniamino Accattoli

Proof nets and the call-by-value λ-calculus

Beniamino Accattoli
INRIA & École Polytechnique (LIX), France

This paper gives a detailed account of the relationship between (a variant of) the call-by-value lambda calculus and linear logic proof nets. The presentation is carefully tuned in order to realize a strong bisimulation between the two systems: every single rewriting step on the calculus maps to a single step on the nets, and viceversa. In this way, we obtain an algebraic reformulation of proof nets. Moreover, we provide a simple correctness criterion for our proof nets, which employ boxes in an unusual way.

1 Introduction

A key feature of linear logic (LL) is that it is a refinement of intuitionistic logic, *i.e.* of λ-calculus. In particular, *one* β-reduction step in the λ-calculus corresponds to the sequence of *two* cut-elimination steps in linear logic, steps which are of a very different nature: the first is multiplicative and the second is exponential. The Curry-Howard interpretation of this fact is that λ-calculus can be refined adding a constructor $t[x/u]$ for *explicit substitutions*, and decomposing a β-step $(\lambda x.t)u \to_\beta t\{x/u\}$ into the sequence $(\lambda x.t)u \to_{\mathtt{m}} t[x/u] \to_{\mathtt{e}} t\{x/u\}$.

Another insight due to linear logic is that proofs can be represented graphically—by the so-called proof nets—and the reformulation of cut-elimination on proof netstakes a quite different flavour with respect to cut-elimination in sequent calculus. The parallel nature of the graphical objects makes the commutative cut-elimination steps, which are the annoying burden of every proof of cut-admissibility, (mostly) disappear.

These two features of LL have influenced the theory of explicit substitutions in various ways [16, 10], culminating in the design of *the structural λ-calculus* [4], a calculus isomorphic (more precisely *strongly bisimilar*) to its representation in LL proof nets [3, 1]. Such a calculus can be seen as an algebraic reformulation of proof nets for λ-calculus [6, 24], and turned out to be simpler and more useful than previous calculi with explicit substitutions.

Girard's seminal paper on linear logic [14] presents two translations of λ-calculus into LL. The first one follows the typed scheme $(A \Rightarrow B)^{\mathtt{n}} = !A^{\mathtt{n}} \multimap B^{\mathtt{n}}$, and it is the one to which the previous paragraphs refer to. It represents the ordinary—or call-by-name (CBN)—λ-calculus. The second one, identified by $(A \Rightarrow B)^{\mathtt{v}} = !(A^{\mathtt{v}} \multimap B^{\mathtt{v}})$, was qualified as *boring* by Girard and received little attention in the literature [21, 23, 8, 11, 12, 20]. Usually, it is said to represent Plotkin's call-by-value (CBV) λ_{β_v}-calculus [22]. These two representations concern typed terms only, but it is well-known that they can be extended to represent the whole untyped calculi by considering linear recursive types ($o = !o \multimap o$ for call-by-name and and $o = !(o \multimap o)$ for call-by-value).

Surprisingly, the extension of the CBV translation to the untyped calculus λ_{β_v}-calculus introduces a violent unexpected behavior: some normal terms in λ_{β_v} map to (recursively typed) proof nets without normal form (see [2] for concrete examples and extensive discussions). This fact is the evidence that there is something inherently wrong in the CBV translation.

To appear in EPTCS.

In this paper we show how to refine the three actors of the play (the CBV λ-calculus, the translation and the proof nets presentation) in order to get a perfect match between terms and proof nets. Technically, we show that the new translation is a strong bisimulation[1], and since strong bisimulations preserve reductions length (in both directions), the normalization mismatch vanishes.

Interestingly, to obtain a strong bisimulation we have to make some radical changes to both the calculus and the presentation of proof nets. The calculus, that we call the *value substitution kernel* λ_{vker} [2], is a subcalculus of the *value substitution calculus* λ_{vsub} studied in [5], which is a CBV λ-calculus with explicit substitutions. Such a kernel is as expressive as the full calculus, and can be thought as a sort of CPS representation of λ_{vsub}.

Here, however, we mostly take the calculus for granted (see [2] for more details) and rather focus on proof nets. Our two contributions are:

1. **Graphical syntax and algebraic formalism**: it is far from easy to realize a strong bisimulation between terms and nets, as it is necessary to take care of many delicate details about weakenings, contractions, representation of variables, administrative reduction steps, and so on. The search for a strong bisimulation may seem a useless obsession, but it is not. Operational properties as confluence and termination then transfer immediately from graphs to terms, and viceversa. More generally, such a strong relationship turns the calculus into an algebraic language for proof nets, providing an handy tool to reason by structural induction over proof nets.
2. **Correctness criterion**: we provide a characterization of the proof nets representing λ_{vker} based on graph-theoretical principles and which does not refer to λ_{vker}, that is, we present a *correctness criterion*. Surprisingly, the known criteria for the representation of the call-by-name λ-calculus (with explicit substitutions) fail to characterize the fragment encoding the call-by-value λ-calculus. Here we present a simple and non-standard solution to this problem. We hack the usual presentation of proof nets so that Laurent's criterion for polarized nets [17, 19, 18]—the simplest known correctness criterion—captures the fragment we are interested in. The hacking of the syntax consists in using boxes for ⅋-links rather than for !-links. An interesting point is that the fragment we deal with is not polarized in Laurent's sense, despite it is polarized in the Lamarche/intuitionistic sense.

The use of boxes for ⅋-links may look terribly ad-hoc. Section 6 tries to argue that it is not. Moreover, Section 7 presents an account of the technical points concerning the representations of terms with proof nets, and how they have been treated in the literature.

2 Terms

In this section we introduce the calculus which will be related to proof nets, called *the value substitution kernel* λ_{vker} [2]. Its syntax is:

$$t,s,u,r \quad ::= \quad x \mid \lambda x.t \mid vs \mid t[x/u] \qquad\qquad v \quad ::= \quad x \mid \lambda x.t$$

where $t[x/u]$ is an *explicit substitution* and values are noted v. Note that the left subterm of an application can only be a value. The rules of λ_{vker} are:

$$(\lambda x.t)u \quad \mapsto_{\mathtt{m}} \quad t[x/u] \qquad\qquad t[x/v\mathtt{L}] \quad \mapsto_{\mathtt{e}} \quad t\{x/v\}\mathtt{L}$$

[1] A strong bisimulation between two rewriting systems S and R is a relation $\equiv$ between S and R s.t. whenever $s \equiv r$ then for every step from $s \to_S s'$ there is a step $r \to_R r'$ s.t. $s' \equiv r'$, *and viceversa* (for $s,s' \in S$ and $r,r' \in R$).

where L is a possibly empty list of explicit substitutions $[x_1/u_1]\dots[x_k/u_k]$ (and the fact that in the lhs of $\mapsto_{\mathsf{e}}$ L appears inside $[\,]$ while in the rhs it appears outside $\{\,\}$ is not a typo). The calculus is confluent [2].

The peculiarity of the value substitution kernel is that iterated applications as $(tu)s$ are not part of the language. The idea is that they are rather represented as $(xs)[x/tu]$ with x fresh. The calculus containing iterated applications is called *the value substitution calculus* λ_{vsub}, and it has been studied in [5, 2]. In [2] it is shown that λ_{vsub} can be represented inside λ_{vker} (mapping iterated applications $(tu)s$ to $(xs)[x/tu]$, as before) and that a term t and its representation t^{k} are equivalent from the point of view of termination (formally t is strongly (resp. weakly) normalizing iff t^{k} is, and the same is true with respect to weak—*i.e.* not under lambda—reduction). If one is interested in observing termination (as it is usually the case) than λ_{vsub} and λ_{vker} are observationally equivalent (via $\cdot^{\mathsf{k}}$). As pointed out to us by Frank Pfenning, the map $\cdot^{\mathsf{k}}$ is reminiscent of the notion of *A-reduction* in the theory of CPS-translations [13, 25]. The idea is then that λ_{vker} (and thus proof nets) is essentially the language of A-normal forms associated to λ_{vsub}. However, the study of the precise relationship with A-normal forms is left to future work.

The calculus λ_{vsub} has been related to Herbelin and Zimmermann's λ_{CBV} [15] in [5]. In turn, λ_{CBV} is related to Plotkin's $\lambda_{\beta v}$ in [15], where it is shown that the equational theory of $\lambda_{\beta v}$ is contained in the theory of λ_{CBV}.

The rest of the paper shows that λ_{vker} can be seen as an algebraic language for the proof nets used to interpret the call-by-value λ-calculus.

3 Proof nets: definition

Introduction. Our presentation of proof nets is non-standard in at least four points (we suggest to have a quick look to Figure 3):

1. **Hypergraphs**: we use hypergraphs (for which formulas are nodes and links—*i.e.* logical rules—are hyperedges) rather than the usual graphs with pending edges (for which formulas are edges and links are nodes). We prefer hypergraphs because in this way contraction can be represented in a better way (providing commutativity, associativity, and permutation with box borders *for free*) and at the same time we can represent cut and axiom links implicitly (similarly to what happens in interaction nets).
2. **⅋-boxes**: We put boxes on ⅋-links and not on !-links. This choice is discussed in Section 6, and it allows to use a very simple correctness criterion—*i.e.* Laurent's criterion for polarized nets—without losing any property.
3. **Polarity**: we apply a polarized criterion to a setting which is not polarized in the usual sense.
4. **Syntax tree**: since we use proof nets to represent terms, we will dispose them on the plane according to the syntax tree of the corresponding terms, and not according to the corresponding sequent calculus proof (also the orientation of the links does not reflect the usual premise-conclusion orientation of proof nets).

Nets. Nets are directed and labelled hyper-graphs $G = (V(G), L(G))$, *i.e.*, graphs where $V(G)$ is a set of labelled **nodes** and $L(G)$ is a set of labelled and **directed hyperedges**, called **links**, which are edges with 0,1 or more sources and 0,1 or more targets[2]. Nodes are labelled with a type in $\{e, m\}$, where e stays for *exponential* and m for *multiplicative*, depicted in blue and brown, respectively. If a node u has

[2] An hyper-graph G can be understood as a bipartite graph B_G, where $V_1(B_G)$ is $V(G)$ and $V_2(B_G)$ is $L(G)$, and the edges are determined by the relations *being a source* and *being a target* of an hyperedge.

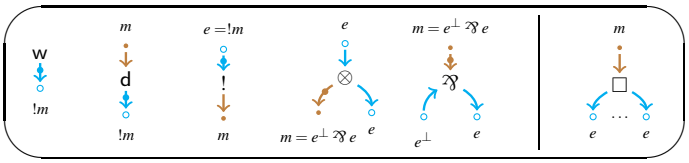

Figure 1: links.

type e (resp. m) we say that it is a e-node (resp. m-node). We shall consider hyper-graphs whose links are labelled from $\{!, \mathsf{d}, \mathsf{w}, ⅋, \otimes\}$. The label of a link l forces the number and the type of the source and target nodes of l, as shown in Figure 1 (the types will be discussed later, and the figure also contains the $\square$-link, which is not used to define nets: it will be used later to define the correction graph). Note that every link (except $\square$) has exactly one connection with a little circle: it denotes the principal node, *i.e.* the node on which the link can interact. Remark the principal node for tensor and !, which is not misplaced. Moreover, every ⅋-link has an associated **box**, *i.e.*, a sub-hyper-graph of G (have a look to Figure 3). The **sources** (resp. **targets**) of a net are the nodes without (resp. outgoing) incoming links; a node which is not a source nor a target is **internal**. Formally:

Definition 3.1 (net)**.** A **net** G is a quadruple $(|G|, B_G, \mathtt{fv}(G), r_G)$, where $|G| = (V(G), L(G))$ is an hyper-graph whose nodes are labelled with either e or m and whose hyperedges are $\{!, \mathsf{d}, \mathsf{w}, ⅋, \otimes\}$-links and s.t.:

- **Root**: $r_G \in V(G)$ is a source e-node of G, called the **root** of G.
- **Conclusions**: $\mathtt{fv}(G)$ is the set of targets of G, also called **free variables** of G, which are targets of $\{\mathsf{d}, \mathsf{w}\}$-links (and not of $\otimes$-links).
- **Multiplicative**: m-nodes have *exactly one* incoming and *one* outgoing link.
- **Exponential**: an e-node has at most one outgoing link, and if it is the target of more than one link then they all are d-links. Moreover, an e-node cannot be isolated.
- **Boxes**: For every ⅋-link l there is a net $box(l)$, called the **box** of l (B_G is the set of boxes of G and $box(l) \in B_G$), with a distinguished free variable x, called the **variable** of l, and s.t.:
  - **Border**: the root $r_{box(l)}$ and the free variable x are the e-nodes of l, and any free variable $\neq x$ of $box(l)$ is not the target of a weakening.
  - **Nesting**: for any two ⅋-boxes $box(l_1)$ and $box(l_2)$ if $\emptyset \neq I := box(l_1) \cap box(l_2)$, $box(l_1) \not\subseteq box(l_2)$, and $box(l_2) \not\subseteq box(l_1)$ then all the nodes in I are free variables of both $box(l_1)$ and $box(l_2)$.
  - **Internal closure**: any link l of G having as target an internal e-node of $box(l)$ is in $box(l)$.
  - **Subnet**: the nodes and the links of $box(l)$ belong to G and the ⅋-links in $box(l)$ inherit the boxes from G.

Some (technical) comments on the definition. In the border condition the fact that the free variables $\neq x$ are not (the target) of a weakening means that weakenings are assumed to be pushed out of boxes as much as possible (of course the rewriting rules will have to preserve this invariant). The internal closure condition is a by-product of collapsing contractions on nodes, which is also the reason of the unusual formulation of the nesting condition: two boxes that are morally disjoint can in our syntax share free variables, because of an implicit contraction merging two of their conclusions.

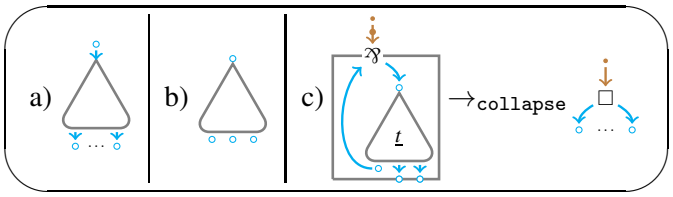

Figure 2: various images.

Terminology about nets. The **level** of a node/link/box is the maximum number of nested boxes in which it is contained[3] (a ⅋-link is not contained in its own box). Two links are **contracted** if they share an e-target. Note that the exponential condition states that only derelictions (*i.e.* d-links) can be contracted. In particular, no link can be contracted with a weakening. A **free weakening** in a net G is a weakening whose node is a free variable of G. Sometimes, the figures show a link in a box having as target a contracted e-node x which is outside the box: in those cases x is part of the box, it is outside of the box only in order to simplify the representation.

Typing. Nets are typed using a recursive type $o = !(o \multimap o)$, that we rename $e = !(e \multimap e) = !(e^{\perp} \parr e)$ because e is a mnemonic for *exponential*. Let $m = e \multimap e = e^{\perp} \parr e$, where m stays for *multiplicative*. Note that $e = !m$ and $m = !m \multimap !m$. Links are typed using m and e, but the types are omitted by all figures except Figure 1 because they are represented using colors and with different shapes (m-nodes are brown and dot-like, e-nodes are white-filled cyan circles). Let us explain the types in Figure 1. They have to be read bottom-up, and thus negated (to match the usual typing for links) if the conclusion of the logical rule is the bottom node of the link, as it is the case for the $\{\mathtt{w}, \mathtt{d}, \otimes\}$-links, while ! and ⅋ have their logical conclusion on the top node, and so their type does not need to be negated.

Induced !*-boxes*. Note that a !-link is always applied to something (m-nodes cannot be conclusions), and there is not so much freedom for this *something*: either it is a dereliction link or a ⅋ with its box. Note also that in both cases we get (what would usually be) a valid content for a !-box. For the dereliction case it is evident, and for the ⅋ case it is guaranteed by the definition of net: the content of a ⅋-box ends on e-nodes. Hence, any !-link has an associated box, induced by ⅋-boxes, which needs not to be represented explicitly.

The translation. Nets representing terms have the general form in Figure 2.a, also schematized as in Figure 2.b. The translation $\underline{\cdot}$ from terms to nets is in Figure 3 (the original boring translation is sketched in Fig. 6, page 12). A net which is the translation of a term is a **proof net**. Note that in some cases there are various connections entering an e-node, that is the way we represent contraction. In some cases the e-nodes have an incoming connection with a perpendicular little bar: it represents an arbitrary number (> 0) of incoming connections. The net corresponding to a variable is given by a ! on a dereliction and not by an (exponential) axiom, as it is sometimes the case. The reason is that an axiom (in our case a node, because axioms are collapsed on nodes) would not reflect on nets some term reductions, as $x[x/v] \to_{\mathtt{e}} v$, for which both the redex and the reduct would be mapped on the same net.

The translation $\underline{\cdot}$ is refined to a translation $\underline{\cdot}_X$, where X is a set of variables, in order to properly handle weakenings during cut-elimination. The reason is that an erasing step on terms simply erases a subterm, while on nets it also introduces some weakenings: without the refinement the translation would not be stable by reduction. The clause defining $\underline{t}_{X \cup \{y\}}$ when $y \notin \mathtt{fv}(t)$ is the first on the second line of Figure 3, the definition is then completed by the following two clauses: $\underline{t}_{\emptyset} := \underline{t}$ and $\underline{t}_{X \cup \{y\}} := \underline{t}_X$ if $y \in \mathtt{fv}(t)$.

[3]Here the words *maximum* and *nested* are due to the fact that the conclusions of ⅋-boxes may belong to two not nested boxes, because of the way we represent contraction.

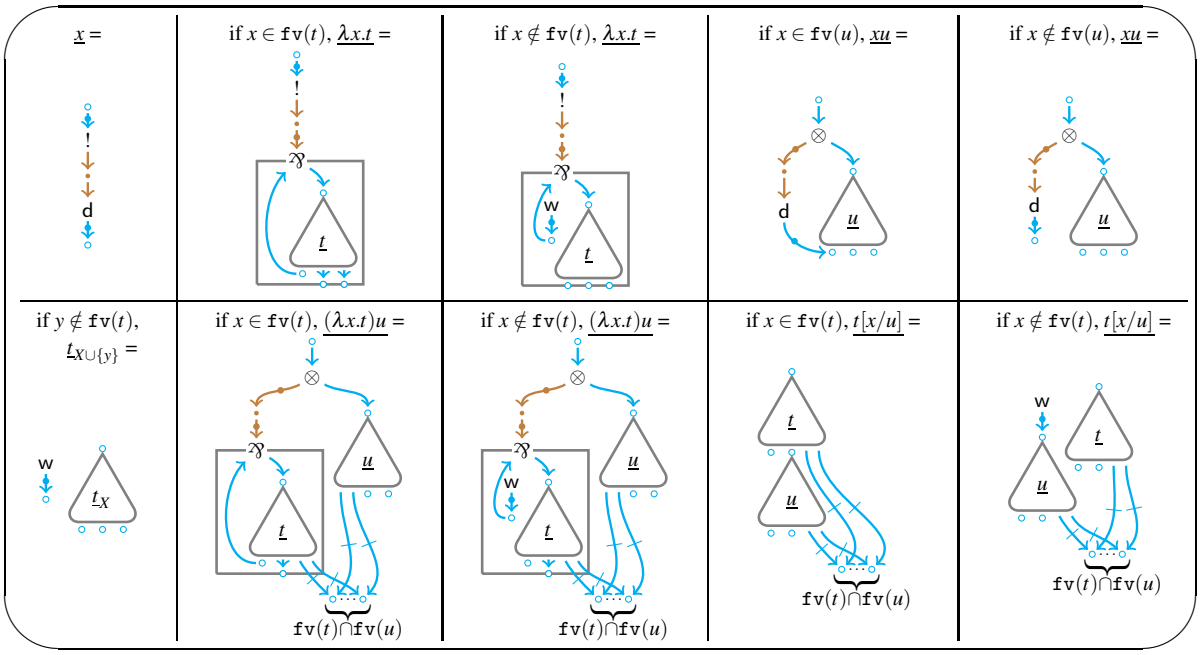

Figure 3: the translation from terms to nets.

α-equivalence. To circumvent an explicit and formal treatment of α-equivalence we assume that the set of *e*-nodes and the set of variable names for terms coincide. This convention removes the need to label the targets of $\underline{t}_X$ with the name of the corresponding free variables in t or X. Actually, before translating a term t it is necessary to pick a *well-named* α-equivalent term t', *i.e.* a term where any two different variables (bound or free) have different names.

Remark 3.2. The translation of terms to nets is not injective. By simply applying the translation it is easily seen that the following pairs of terms have the same net:

$$\begin{array}{llll} t[x/s][y/u] & \sim_{vo_{CS}} & t[y/u][x/s] & \text{if } x \notin \mathtt{fv}(u) \ \&\ y \notin \mathtt{fv}(s) \\ v\ u[x/s] & \sim_{vo_1} & (v\ u)[x/s] & \text{if } x \notin \mathtt{fv}(v) \\ t[x/s[y/u]] & \sim_{vo_2} & t[x/s][y/u] & \text{if } y \notin \mathtt{fv}(t) \end{array} \tag{1}$$

Let $\equiv_{vo}$ be the reflexive, transitive, and contextual closure of $\sim_{vo_{CS}} \cup \sim_{vo_1} \cup \sim_{vo_2}$. In the proof of Lemma 5.1, we will use the fact that if $t \equiv_{vo} s$ then t and s are mapped on the same net. We also claim—without proving it—that $\equiv_{vo}$ is exactly the quotient induced on terms by the translation to nets.

Paths. A path τ of length $k \in \mathbb{N}$ from u to v, noted $\tau : u \to^k v$, is an alternated sequence $u = u_1, l_1, \ldots, l_k, u_{k+1} = v$ of nodes and links s.t. the link l_i has source u_i and target u_{i+1} for $i \in \{1, \ldots, k\}$. A cycle is a path $u \to^k u$ with $k > 0$.

Correctness. The correctness criterion is based on the notion of correction graph, which is—as usual for nets with boxes—obtained by collapsing every box at level 0 into a generalized axiom link.

Definition 3.3 (correction graph)**.** Let G be a net. The correction graph G^0 of G is the hyper-graph obtained from G by collapsing any $\parr$-box at level 0 into a □-link applying the rule in Fig. 2.c.

Definition 3.4 (correctness)**.** A net G is correct if:

- **Source**: G^0 has exactly one *e*-source (the root of G).

- **Acyclicity**: G^0 is acyclic.
- **Recursive correctness**: the interior of every box is correct.

As usual an easy induction on the translation shows that the translation of a term is correct, *i.e.* that:

Lemma 3.5. *Every proof net is correct.*

4 Proof nets: sequentialization

In this section we show how to extract a term t from every correct net G in such a way that t translates back to G, *i.e.* we show that every correct net is a proof net. The proof of this fact is based on the notion of *kingdom*, along the lines of the proof for polarized nets, see [18] (pp. 57-63).

Definition 4.1 (Kingdom)**.** Let G be a correct net and $x \notin \mathtt{fv}(G)$ one of its e-nodes. The **kingdom** $king(x)$ of x is the set of links defined by induction on the link l of source x:

- l is a !-link: $king(x)$ is given by l plus the $\mathtt{d}$-link or the $\parr$-box on the m-target of l.
- l is a $\otimes$-link: $king(x)$ is given by l plus the $\mathtt{d}$-link or the $\parr$-box on the m-target of l plus $king(y)$, where y is the e-target of l.

The main property of $king(x)$ is that it is the smallest subnet of root x, as we shall soon prove[4]. To state this fact precisely we need the notion of subnet.

Definition 4.2 (subnet)**.** Let G be a correct net. A subnet H of G is a subset of its links s.t. it is a correct net and satisfying:

- **Internal closure**: if x is an internal e-node of H then any link of G of target x belongs to H.
- **Box closure**:
  - **Root**: if a $\parr$-link l belongs to H then its box does it too.
  - **Free variables**: if a free variable of a box B of G is internal to H then $B \subseteq H$.

The following lemma is essentially obvious, and usually omitted, but in fact it is used in the proof of Lemma 4.5.

Lemma 4.3. *Let G be a correct net, H a subnet of G, x an internal e-node of H. Then there exists a subnet K of H having x as root and s.t. it is a subnet of G.*

Proof. It is enough to show that there is a subnet of H of root x, since it is obvious that any subnet of K is a subnet of G. By induction on the length of the maximum path from x to a free variable of K. □

To properly describe kingdoms we need the following definition.

Definition 4.4 ((free/ground) substitution)**.** Let G be a correct net. A **substitution** is an e-node which is the target of a $\{\mathtt{w},\mathtt{d}\}$-link (or, equivalently, which is not the target of a $\otimes$-link) and the source of some link. A substitution x is **ground** if it is a node of G^0 (*i.e.* it is not internal to any $\parr$-box[5]), and it is **free** if it is ground and there is no ground substitution of G to which x has a path (in G^0).

Lemma 4.5 (kingdom)**.** *Let G be a correct net and $x \notin \mathtt{fv}(G)$ one of its e-nodes. $king(x)$ is the kingdom of x,* i.e.*, the smallest subnet of G rooted at x. Moreover, it has no free substitutions, no free weakenings, and whenever $y \in \mathtt{fv}(king(x))$ is internal to a subnet H of G then $king(x) \subseteq H$.*

[4] We call *kingdom of x* the net in def. 4.1, but at this point nothing guarantees that it is the smallest subnet of root x.

[5] Note that our collapsed representation of contractions and cuts does not allow to simply say that x is a node at level 0: indeed the conclusion of a $\parr$-box can have level > 0 and yet belong to G^0.

Proof. Let H be a correct subnet of G rooted at x. We show by induction on the length of the maximum path from x to a free variable of G that $king(x) \subseteq H$ and that $king(x)$ is correct. Let l be the link of source x. Cases:

- **Base case**: l is a !-link. By the conclusion condition H has to contain the d-link i or the $\parr$-link on the m-target of l. In the case of a $\parr$-link the box closure condition implies that the whole box B is in H, hence $king(x) \subseteq H$. In the case of a d-link correctness is obvious, in the case of a $\parr$-box it follows by the correctness of the interior of the box, guaranteed by the recursive correctness condition. Moreover, no free substitutions and no free weakenings belong to $king(x)$ (boxes cannot close on weakenings). Pick $y \in \mathtt{fv}(king(x))$, which in the d-link case is the target of i and in the other case is a free variable of the $\parr$-box B. If y is internal to H then the conditions for a subnet guarantee that i or B are in H. Then clearly $king(x) \subseteq H$.
- **Inductive case**: l is a $\otimes$-link. As in the previous case H has to contain the d-link or the $\parr$-box on the m-target of l. Moreover, by lemma 4.3 H contains a subnet K rooted in the e-target y of l. By inductive hypothesis $king(y)$ is the kingdom of y, therefore we get $king(y) \subseteq K \subseteq H$. Hence $king(x) \subseteq H$. By *i.h.* we also get that $king(y)$ is correct, hence y is its only e-source and x is the only e-source of $king(x)$. Acyclicity follows by correctness of G. Recursive correctness follows from the box closure condition and correctness of G. Moreover, by *i.h.* $king(y)$—and so $king(x)$—has no free substitutions and no free weakenings. The part about free variables uses the *i.h.* for the free variables of $king(y)$ and the conditions for a subnet as in the previous case for the other free variables. □

Lemma 4.6 (substitution splitting)**.** *Let G be a correct net with a free substitution x. Then*

1. *The free variables of $king(x)$ are free variables of G.*
2. *$G \setminus king(x)$ is a subnet of G.*

Proof. 1) Suppose not. Then there is a free variable y of $king(x)$ which is not a free variable of G. There are two possible cases:

- *y is a substitution*. Then x has a path to a substitution in G^0, against the definition of free substitution, absurd.
- *y is the distinguished free variable of a $\parr$-box B*. Thus, y is internal to some $\parr$-box B and so it is not a node of G^0. By Lemma 4.5 we get that $king(x) \subseteq B$ and so x is not a node of G^0, against the definition of free substitution, absurd.

2) By point 1 the removal of $king(x)$ cannot create new e-sources. Being a substitution, x is the target of some link. Therefore the removal of $king(x)$ cannot remove the root of G. It is also clear that the removal cannot create cycles, and the box closure condition for subnets guarantees that the recursive correctness of G implies the one of $G \setminus king(x)$. □

Lemma 4.7. *Let G be a correct net with a ground substitution. Then G has a free substitution.*

Proof. Consider the following order on the elements of the set S_g of ground substitutions of G: $z \leq y$ if there is a path from z to y in G^0. Acyclicity of G^0 implies that S_g contains maximal elements with respect to $\leq$, if it is non-empty. Note that a maximal element of S_g is a free substitution in G. Now, if G has a ground substitution x then S_g is non-empty. Thus, G has a free substitution. □

The next lemma is used in the proof of the sequentialization theorem.

Lemma 4.8 (kingdom characterization)**.** *Let G be a correct net. Then $G = king(r_G)$ iff G has no free substitutions nor free weakenings.*

Proof. $\Rightarrow$) By Lemma 4.5. $\Leftarrow$) By lemma 4.5 we get that $king(r_G) \subseteq G$. If the two do not coincide then by the internal closure condition for subnets, the multiplicative condition on nets, and the fact that they share the same root, we get that G contains a ground substitution x on a free variable of $king(r_G)$. By lemma 4.7 G contains a free substitution, absurd. □

Theorem 4.9 (sequentialization)**.** *Let G be a correct net and X be the set of e-nodes of its free weakenings. Then there is a term t s.t. $\underline{t}_X = G$ (and $\mathtt{fv}(G) = \mathtt{fv}(t) \cup X$).*

Proof. By induction on the number of links. By the root and conclusion conditions the minimum number of links is 2 and the two links are necessarily a !-link on top of a d-link. Let x be the e-node of the d-link. Then $\underline{x} = G$. We now present each inductive case. After the first one we assume that the net has no free weakening.

- *There is a free weakening l of e-node y.* Then $G' = G \setminus \{l\}$ is still a correct net and by *i.h.* there exist t s.t. $\underline{t}_{X\setminus\{y\}} = G'$. Then $\underline{t}_X = G$.
- *There is a free substitution x.* Then by Lemma 4.5 and Lemma 4.6 $king(x)$ and $G \setminus king(x)$ are correct subnets of G. By the *i.h.* there exist s and u s.t. $\underline{s} = king(x)$ and $\underline{u}_{\{x\}} = G \setminus king(x)$ (note that if $x \in \mathtt{fv}(u)$ then $\underline{u}_{\{x\}} = \underline{u}_{\emptyset} = \underline{u}$). Then $\underline{u[x/s]} = G$.
- *No free substitution*: by lemma 4.8 $G = king(r_G)$. In case the root link l of G is:
  - *a !-link over a d-link*: base case, already treated.
  - *a !-link over a $\parr$-link*: let H be the box of the $\parr$-link and x its distinguished free variable. By definition of a net the set of free weakenings of H either is empty or it contains only x. If x is (resp. is not) the node of a free weakening then by *i.h.* there exists t s.t. $\underline{t}_{\{x\}} = H$ (resp. $\underline{t} = H$). Then $\underline{\lambda x.t} = G$.
  - *A $\otimes$-link l*: let x be its e-target and a its m-target. Note that $G = king(r_G)$ implies that G is composed by l, $king(x)$ and either the d-link or the $\parr$-link (plus its box) on a. By *i.h.* there exists s s.t. $\underline{s} = king(x)$. Now, if a is the source of a d-link of e-node y we conclude, since $\underline{ys} = G$. Otherwise, s is the source of a $\parr$ of box H and the *i.h.* gives a term u and a set X s.t. $\underline{u}_X = H$. Let us prove that H and $king(x)$ can only share free variables, as the translation prescribes: no link at level 0 of $king(x)$ can be in H, and no box at level 0 of $king(x)$ can intersect H other than on free variables, by the nesting condition. By reasoning about the distinguished free variable of H as in the previous case we then get $\underline{(\lambda y.u)s} = G$. □

5 Proof nets: dynamics

The rewriting rules are in Figure 4. Let us explain them. First of all, note that the notion of cut in our syntax is implicit, because cut-links are not represented explicitly. A cut is given by a node whose incoming and outgoing connections are principal (*i.e.* with a little square on the line).

The rule $\to_{\mathtt{m}}$ is nothing but the usual elimination of a multiplicative cut, except that the step also opens the box associated with the $\parr$-link.

The two $\to_{\mathtt{e}}$ rules reduce the exponential redexes. Let us explain how to read them. For the graph noted H in Figure 4 there are two possibilities: either it is simply a dereliction link (a d-link) or it is a $\parr$ with its box, so there is no ambiguity on what to duplicate/erase. Every pair of short gray lines denotes

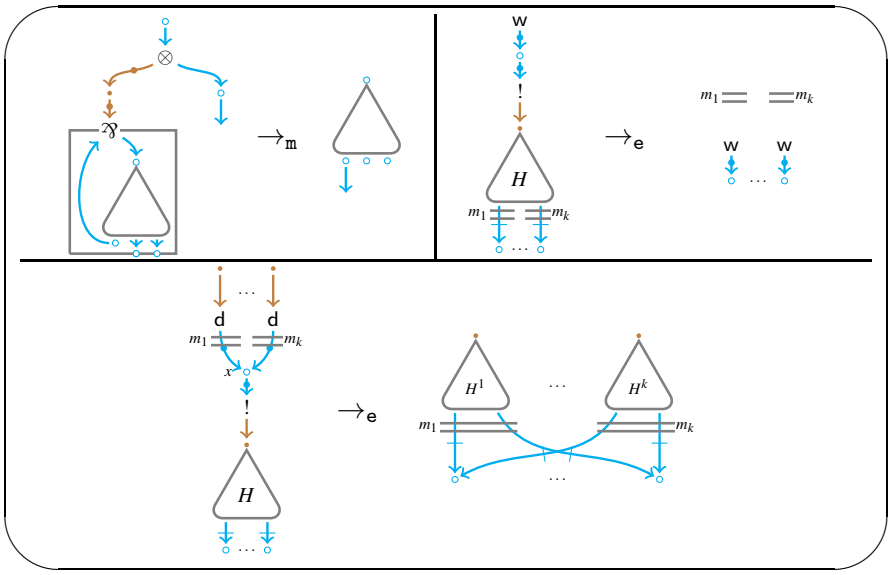

Figure 4: proof nets cut-elimination rules

the sequence (of length m_i, with $i \in \{1,\dots,k\}$) of boxes closing on the corresponding links. The rule has two cases, one where ! is cut with $k \in \{1,2,\dots\}$ derelictions and one where it is cut with a weakening. In the first case the sub-graph H is copied k times (if $k=1$ no copy is done) into $H^1,\dots H^k$ and each copy enters in the m_i boxes enclosing the corresponding (and removed) dereliction. Moreover, the k copies of each target of H are contracted together, *i.e.* the nodes are merged. In the case of a cut with a weakening, H is erased and replaced by a set of weakenings, one for every target of H. Note that the weakenings are also pushed out of all boxes closing on the targets of H[6]. This is done to preserve the invariant that weakening are always pushed out of boxes as much as possible. Such invariant is also used in the rule: the weakening is at the same level of H. Last, if the weakenings created by the rule are contracted with any other link then they are removed on the fly (because by definition weakenings cannot be contracted).

Now, we establish the relationship between terms and nets at the level of reduction. Essentially, there is only one fact which is not immediate, namely that $\to_{\mathtt{e}}$ actually implements the $\to_{\mathtt{e}}$ rule on terms, as it is proved by the following lemma.

Lemma 5.1 (substitution)**.** *Let* $t = s[x/v\mathtt{L}]$ *then* $\underline{t}_X \to_{\mathtt{e}} \underline{s\{x/v\}\mathtt{L}}_X$ *for any set of names* $X \supseteq \mathtt{fv}(t)$.

Proof. First of all observe that t and $s[x/v]L$ both reduce to $s\{x/v\}L$ and by remark 3.2 both translate to the same net. Hence it is enough to prove that $\underline{s[x/v]L}_X \to_e \underline{s\{x/v\}L}_X$. We prove it by induction on the number k of substitutions in L. If $k=0$ then the proof is by induction on the number n of free occurrences of x in s. Cases:

- $n=0$) In $\underline{s[x/v]}_X$ the bang associated to v is cut with a weakening. The elimination of the cut gets a net G' without the !-link and the ⅋-box associated to v, leaving a free weakening for every free variable of the box, *i.e.* of every free variable of v: then G' is exactly $\underline{s\{x/v\}}_{X\cup\mathtt{fv}(v)} = \underline{s}_{X\cup\mathtt{fv}(v)}$.
- $n>1$) Write $s = C[x]$ for some occurrence of x. Now, consider $u = C[y][y/v][x/v]$ and note that:

[6]Note that, for the sake of a simple representation, the figure of the weakening cut-elimination rule is slightly wrong: it is not true that the links $l_1,\dots,l_j$ having as target a given conclusion x_i of H are all inside m_i boxes, because each one can be inside a different number of boxes.

$$u \to C[v][x/v] \to C[v]\{x/v\} = s\{x/v\}$$

The difference between $G' = \underline{u}_X$ and $G = \underline{s[x/v]}_X$ is that one of the occurrences of x in G has been separated from the others and cut with a copy of $\underline{v}$. Consider the step $G \to H$ which reduces the cut on x in G and the sequence $G' \to H'_y \to H'_{y,x}$ which first reduces the cut on y in G' and then reduces in H' the (unique) residual of the cut on x in G'. By the definition of reduction in nets $H = H'_{y,x}$. Now by *i.h.* applied to u and y we get that $\underline{C[v][x/v]}_X = H'_y$ and by the *i.h.* applied to $C[v][x/v]$ and x we get that $\underline{C[v]\{x/v\}}_X = H'_{y,x}$. From $H = H'_{y,x}$ and $C[v]\{x/v\} = s\{x/v\}$ we get $\underline{s\{x/v\}}_X = H$ and conclude.

- $n = 1$) By induction on s. Some cases:
  - If $t = \lambda y.u$ then by *i.h.* $\underline{u[x/v]}_{X\cup\{y\}} \to_e \underline{u\{x/v\}}_{X\cup\{y\}}$ and so we get $\underline{\lambda y.(u[x/v])}_{X\cup\{y\}} \to_e \underline{\lambda y.(u\{x/v\})}_{X\cup\{y\}}$. Now, observe that $\lambda y.(u\{x/v\}) = (\lambda y.u)\{x/v\} = t\{x/v\}$ and that the two nets $\underline{\lambda y.(u[x/v])}_{X\cup\{y\}}$ and $\underline{(\lambda y.u)[x/v]}_{X\cup\{y\}}$ have the same reduct after firing the exponential cut on x, and so we get $\underline{(\lambda y.u)[x/v]}_{X\cup\{y\}} \to_e \underline{(\lambda y.u)\{x/v\}}_{X\cup\{y\}}$.
  - If $s = w[y/u]$ then either $x \in u$ or $x \in w$. In the first case by remark 3.2 we get that $\underline{s[x/v]}_X = \underline{w[y/u][x/v]}_X = \underline{w[y/u[x/v]]}_X$. Now by *i.h.* $\underline{u[x/v]} \to_e \underline{u\{x/v\}}$. Then we have $\underline{s[x/v]}_X \to_e \underline{w[y/u\{x/v\}]}_X = \underline{w[y/u]\{x/v\}}_X = \underline{s\{x/v\}}_X$. The second case is analogous.
  - If $s = (\lambda y.w)u$. The case $x \in u$ uses remark 3.2 and the *i.h.* as in the $s = w[y/u]$ case. The case $x \in w$ is slightly different. As before $((\lambda y.w)u)[x/v]$ and $((\lambda y.w[x/v])u)$ have the same reduct. By *i.h.* hypothesis $\underline{w[x/v]} \to_e \underline{w\{x/v\}}$ and thus $\underline{(\lambda y.w[x/v])u}_X \to_e \underline{(\lambda y.w\{x/v\})u}_X$. We conclude since $\underline{((\lambda y.w)u)[x/v]}_X \to_e \underline{((\lambda y.w\{x/v\})u)}_X = \underline{((\lambda y.w)u)\{x/v\}}_X$.

If $k > 0$ and $\mathtt{L} = \mathtt{L}'[y/r]$ then we get by *i.h.* that $\underline{s[x/v]\mathtt{L}'}_X \to_e \underline{s\{x/v\}\mathtt{L}'}_X$. By definition of the translation and of graph reduction it follows that $\underline{s[x/v]\mathtt{L}'[y/r]}_X \to_e \underline{s\{x/v\}\mathtt{L}'[y/r]}_X$. □

Theorem 5.2 (strong bisimulation)**.** *Let t be a term and X a set of variables containing $\mathtt{fv}(t)$. The translation is a strong bisimulation between t and $\underline{t}_X$,* i.e. *$t \to_a t'$ if and only if $\underline{t}_X \to_a \underline{t'}_X$, for $a \in \{\mathtt{m}, \mathtt{e}\}$.*

Proof. By induction on the translation. If $t = x$ there is nothing to prove, and if $t = \lambda x.s$ or $t = xs$ it immediately follows by the *i.h.*, since all the redexes of t are contained in s. If $t = s[x/u]$ and the redex is in s or u then just apply the *i.h.*. If $u = v\mathtt{L}$ and the redex is $s[x/v\mathtt{L}] \to_{\mathtt{e}} s\{x/v\}\mathtt{L}$ then apply Lemma 5.1. If $t = (\lambda x.s)u$ and the redex is in s or u then just apply the *i.h.*. If $t = (\lambda x.s)u \to_{\mathtt{m}} s[x/u] = t'$ then have a look at Figure 5.a: clearly $t \to_{\mathtt{m}} t'$ iff $\underline{t}_X \to_{\mathtt{m}} \underline{t'}_X$. □

Strong bisimulations preserve reduction lengths, so they preserve divergent/normalizing reductions, and termination properties in general.

Technical digression about confluence. For confluence the point is slightly more delicate, since in general it is preserved only modulo the quotient induced by the strong bisimulation. But mild additional hypothesis allow to transfer confluence. Given two rewriting systems $(S_1, \to)$ and $(S_2, \rightsquigarrow)$ and a strong bisimulation $\equiv$ (defined on all terms of S_1 and S_2), to transfer confluence from S_1 to S_2 it is enough to ask that if $s_1 \equiv s_2$ and $s_1 \to s'_1$ then there is a unique s'_2 s.t. $s_2 \rightsquigarrow s'_2$ and $s_2 \equiv s'_2$, see [1] (pp. 83-86) for more details. It is easily seen that in our case the translation enjoys this property in both directions.

These observations (and confluence of λ_{vker}) prove:

Corollary 5.3. *Let $t \in \lambda_{vker}$ and X a set of variables. Then t is weakly normalizing/strongly normalizing/a normal form/without a normal form iff $\underline{t}_X$ is. Moreover, proof nets are confluent.*

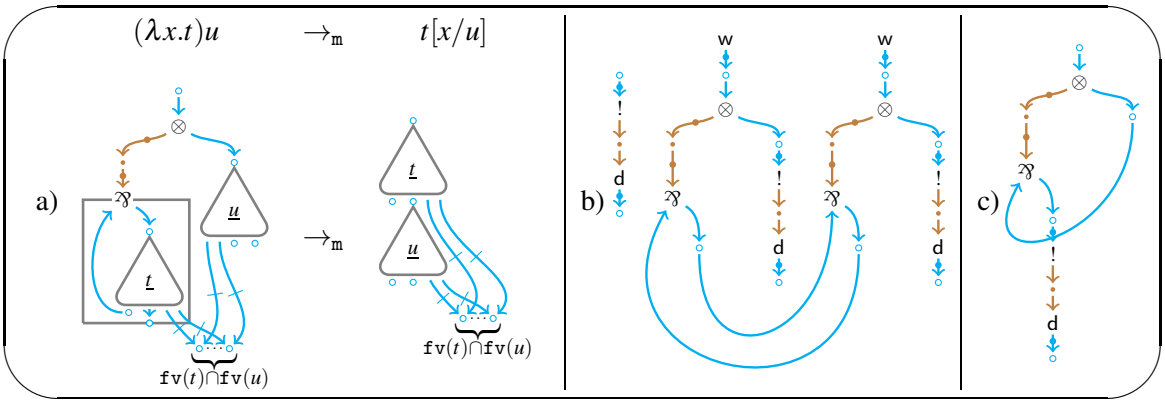

Figure 5: a) A $\to_{\mathtt{m}}$-step on terms and on nets. b-c) Counter-examples to correctness without $\mathfrak{Y}$-boxes

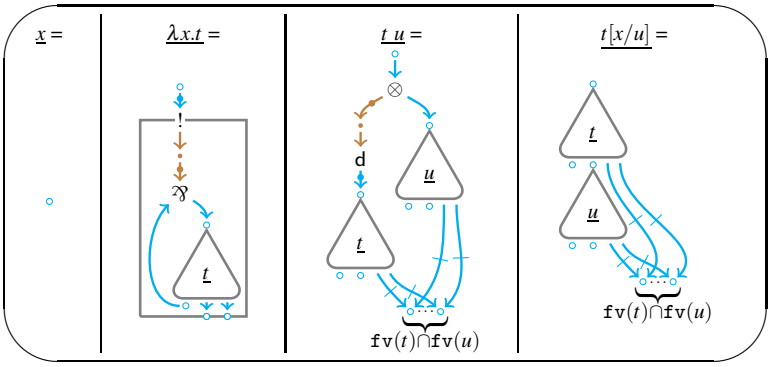

Figure 6: the ordinary CBV translation from terms to nets.

Actually, the translation is more than a strong bisimulation: the reduction graphs[7] of t and $\underline{t}$ are *isomorphic*, not just strongly bisimilar. An easy but tedious refinement of the proof of Theorem 5.2 proves:

Theorem 5.4 (dynamic isomorphism)**.** *Let t be a term and X a set of variables containing $\mathtt{fv}(t)$. The translation induces a bijection ϕ between the redexes of t and the redexes of $\underline{t}_X$ s.t. $R : t \to_a t'$ if and only if $\phi(R) : \underline{t}_X \to_a \underline{t'}_X$, where $a \in \{\mathtt{m}, \mathtt{e}\}$.*

A nice by-product of the strong bisimulation approach is that preservation of correctness by reduction *comes for free*, since any reduct of a proof-net is the translation of a term.

Corollary 5.5 (preservation of correctness)**.** *Let G be a proof net and $G \to G'$. Then G' is correct.*

The original boring translation. For the sake of completeness, Figure 6 sketches the ordinary CBV translation from λ-terms (possibly with iterated applications) to proof nets (including the case for explicit substitutions and using a traditional syntax with boxes on !). An easy computation shows that the term $t = \delta(yz)\delta$, where $\delta = \lambda x.xx$ maps to a net without normal form, while t is a $\lambda_{\beta v}$-normal form (see [2] for more details). This mismatch is the motivation behind our work.

[7] *Reduction graphs*, which are the graphs obtained considering all reductions starting from a given object, *are not nets*.

6 Motivating $\parr$-boxes

The two encodings of λ-calculus can be seen as fragments of Intuitionistic Multiplicative and Exponential Linear Logic (IMELL). Let us stress that in IMELL what we noted $\otimes$ and $\parr$ correspond to the right and left rules for the linear implication $\multimap$, and not to the left and right rules for $\otimes$ (the four rules for $\otimes$ and $\multimap$ are collapsed in LL but not in Intuitionistic LL, in particular our $\parr$ acts on the output of the term, *i.e.* on the right of the sequent, and corresponds to the right rule for $\multimap$).

Our argument is that in IMELL there is no correctness criterion unless the syntax is extended with boxes for both ! *and* $\multimap$ (our $\parr$), as we shall explain in the next paragraphs. The fragment of IMELL encoding the CBN λ-calculus is a special case where the box for $\multimap$ needs not to be represented. The fragment encoding the CBV λ-calculus is a special case where the box for ! needs not to be represented. So, the two encodings are dual with respect to the use of boxes, and then there is nothing exotic in our use of $\parr$-boxes.

The difficulty of designing a correctness criterion for IMELL is given by the presence of weakenings, which break connectedness. In most cases weakenings simply prevent the possibility of a correctness criterion. The fragment encoding the CBN λ-calculus, and more generally Polarized Linear Logic, are notable exceptions. For the encoding of the CBN λ-calculus there exist two correctness criteria. Let us show that none of them works for the CBV λ-calculus.

The first is the Danos-Regnier criterion, in the variant replacing connectedness with the requirement that the number of connected components of every switching graph is $1 + \#w$, where $\#w$ is the number of weakenings at level 0 (after the collapse of !-boxes) [24]. In our case this criterion does not work: the net in Fig. 5.b verifies the requirement while it does not represent any proof or term. The second criterion is Olivier Laurent's polarized criterion, because the CBN encoding is polarized. In its original formulation it cannot be applied to the encoding of the CBV λ-calculus, because such a fragment is not polarized (there can be a weakening as a premise of a tensor, which is forbidden in polarized logic). Our re-formulation of Laurent's criterion rejects the net in Figure 5.b (because the two $\parr$-links form a cycle), but without using $\parr$-boxes it would accept the net in Figure 5.c, which is not correct[8].

Thus, the known criteria do not work and there is no criteria for IMELL. The usual way to circumvent problems about correctness is to add some information to the graphical representation, under the form of boxes (as we did) or jumps (*i.e.* additional connections). It is well known that in these cases various criteria can be used, but this extra information either is not canonical or limits the degree of parallelism. Another possible solution is to modify the logical system adding the mix rules. However, such rules are debatable, and also give rise to a bad notion of subnet (for details see [1], pp. 199-201).

Let us stress that our counter-examples to the known criteria do not rely on the exponentials (*i.e.* non-linearity): it is easy to reformulate them in Intuitionistic Multiplicative Linear Logic (IMLL) with units[9], for which then there is no correctness criterion.

In the case studied in this paper the use of $\parr$-boxes does not affect the level of parallelism in a sensible way. Indeed, in IMELL the parallelism given by proof nets concerns the left rules (of $\otimes$ and $\multimap$, plus contractions and weakenings) and cuts: in our case there is no $\otimes$ (remember our $\otimes$ and $\parr$ rather correspond to the rules for $\multimap$), our technical choices for variables keep the parallelism for contraction and weakenings, and the parallelism of the left rule for $\multimap$ (our $\otimes$) and cuts is preserved (it is given by the equations in (1), page 6).

[8] The net in Figure 5.c would be rejected by the original version of the criterion, which is based on a different orientation. But the original orientation cannot be applied to our fragment.

[9] Just replace each sequence of a ! over a dereliction with an axiom, and the weakenings with $\bot$-links.

7 Proof nets: the literature on term representations

When relating λ-terms and proof nets a number of technical choices are possible:

1. *Explicit substitutions*: proof nets implement a β-step by two cut-elimination steps. This refined evaluation can be seen on the calculus only if the syntax is extended with explicit substitutions.
2. *Variables*: to properly represent variables it is necessary to work modulo associativity and commutativity of contractions, neutrality of weakening with respect to contraction, and permutations of weakenings and contractions with box-borders. In the literature there are two approaches: to explicitly state all these additional congruences or to use a syntax naturally quotienting with respect to them. Such a syntax uses n-ary ?-links collapsing weakening, dereliction and contractions and delocalizing them out of boxes. It is sometimes called *nouvelle syntaxe*.
3. *Axioms*: various complications arise if proof nets are presented with explicit axiom and cut links. They can be avoided by working modulo cuts on axioms, which is usually done by employing an interaction nets presentation of proof nets.
4. *Exponential cut-elimination*: the cut-elimination rules for the exponentials admit many presentations. Essentially, either they are big-step, *i.e.* an exponential cut is eliminated in one shot (making many copies of the !-premise of the cut), or they are small-step, with a rule for each possible ?-premise (weakening, dereliction, contraction, axiom, box auxiliary port).

We now list the works in the literature which are closer in spirit to ours, *i.e.* focusing on the representation of λ-calculi into proof nets (and for space reasons we omit many other interesting works, as for instance [20], which studies the representation of *strategies*, not of *calculi*). The first such works were the Ph.D. thesis of Vincent Danos [6] and Laurent Regnier [24], which focused on the call-by-name (CBN) translation. Danos and Regnier avoid explicit substitutions, use n-ary contractions, explicit axioms, and big-step exponential rules, see also [7]. They characterize the image of the translation using the variant on the Danos-Regnier correcteness criterion which requires that any switching graph has $\#w+1$ connected components, where $\#w$ is the number of weakenings. In [8] Danos and Regnier use the CBV translation[10]. Both translations are injective.

In [19, 18] Olivier Laurent extends the CBN translation to represent (the CBN) $\lambda\mu$-calculus. He does not use explicit substitutions nor n-ary ?-links, while he employs explicit axiom links and small-step exponential rules. His work presents two peculiar points. First, the translation of $\lambda\mu$-terms is not injective, because—depending on the term—the μ-construct may have no counterpart on proof nets. This induces some mismatches at the dynamic level. Second, Laurent finds a simpler criterion, exploiting the fact that the fragment encoding (the CBN) $\lambda\mu$-calculus is polarized. In [18] Laurent also show how to represent the CBV $\lambda\mu$-calculus. However, such a representation does not use the same types of the boring translation, as $A \to B$ maps to $?!(A \multimap B)$, and not to $!(A \multimap B)$.

Lionel Vaux [28] and Paolo Tranquilli [26, 27] study the relationship between the differential λ-calculus and differential proof nets. Vaux also extends the relationship to the classical case (thus encompassing a differential $\lambda\mu$-calculus), while Tranquilli refines the differential calculus into a *resurce calculus* which better matches proof nets. They do not use explicit substitutions, nor n-ary contractions, while they use interaction nets (so no explicit axioms and cut link) and small-step exponential rules. Both Tranquilli and Vaux rely on the Danos-Regnier criterion, despite the fragment encoding their calculi is

[10] Let us point out that [8] presents an oddity that we believe deserves to be clarified. The authors show that an optimized geometry of interaction for the proof nets of the CBV-translation is isomorphic to Krivine' s abstract machine (KAM): this is quite puzzling, because the KAM is CBN, while they use the CBV translation.

polarized and can be captured using Laurent's criterion by using boxes for coderelictions; in the context of λ-calculus such boxes do not reduce the parallelism of the representation.

Delia Kesner and co-authors [9, 10, 16] study the relationship with explicit substitutions (in the CBN case). The main idea here is that explicit substitutions correspond to exponential cuts. They use explicit axiom links and small-step exponential rules, but they do not employ n-ary contractions (and so they need additional rules and congruences). Because of explicit substitutions the translation is not injective: now different terms may map to the same proof net, as in this paper. They do not deal with correctness.

In none of these works the translation is a strong bisimulation. In [3] the author and Stefano Guerrini use a syntax inspired by proof nets (and extended with jumps) to represent the CBN λ-calculus with explicit substitutions. That work is the only one employing (the equivalent of) n-ary ?-links and (the equivalent of) small-step exponential rules. In [3] the correctness criterion is a variation over Lamarche's criterion for essential nets, which relies in an essential way on the use of jumps. A reformulation in the syntactic style of this paper of both [3] and of Danos and Regnier's proof nets for the CBN λ-calculus can be found in [1], together with a detailed account of the strong bisimulation.

Here, hypergraphs allow us to use n-ary ?-links and collapse axioms and cut links (as if we were using interaction nets). More precisely, we represent n-ary ?-links by allowing *e*-nodes to have more than one incoming link. This choice overcomes some technicalities about *gluing* and *de-gluing* of ?-links. Such technicalities are always omitted, but they are in fact necessary to properly define subnets and cut-elimination. We also employ big-step exponential rules and explicit substitutions.

Acknowledgements. To Stefano Guerrini, for introducing me to proof nets, correctness and the representation of λ-terms, and to Delia Kesner, for helping with the financial support of this work.

References

[1] Beniamino Accattoli (2011): *Jumping around the box: graphical and operational studies on λ-calculus and Linear Logic*. PhD thesis, *La Sapienza* University of Rome.

[2] Beniamino Accattoli (2012): *A linear analysis of call-by-value λ-calculus*. Available at the address `https://sites.google.com/site/beniaminoaccattoli/cbv-analysis.pdf?attredirects=0`.

[3] Beniamino Accattoli & Stefano Guerrini (2009): *Jumping Boxes*. In: *CSL*, pp. 55–70. Available at `http://dx.doi.org/10.1007/978-3-642-04027-6_7`.

[4] Beniamino Accattoli & Delia Kesner (2010): *The Structural λ-Calculus*. In: *CSL*, pp. 381–395. Available at `http://dx.doi.org/10.1007/978-3-642-15205-4_30`.

[5] Beniamino Accattoli & Luca Paolini (2012): *Call-by-Value Solvability, revisited*. In: *FLOPS*, pp. 4–16. Available at `http://dx.doi.org/10.1007/978-3-642-29822-6_4`.

[6] Vincent Danos (1990): *La Logique Linéaire appliqué à l'étude de divers processus de normalisation (principalement du λ-calcul)*. Phd thesis, Université Paris 7.

[7] Vincent Danos & Laurent Regnier (1995): *Proof-nets and the Hilbert space*. In: *Advances in Linear Logic*, Cambridge University Press, pp. 307–328. Available at `http://dx.doi.org/10.1017/CBO9780511629150.016`.

[8] Vincent Danos & Laurent Regnier (1999): *Reversible, Irreversible and Optimal lambda-Machines*. *Theor. Comput. Sci.* 227(1-2), pp. 79–97. Available at `http://dx.doi.org/10.1016/S0304-3975(99)00049-3`.

[9] Roberto Di Cosmo & Delia Kesner (1997): *Strong Normalization of Explicit Substitutions via Cut Elimination in Proof Nets (Extended Abstract)*. In: *LICS*, pp. 35–46. Available at `http://doi.ieeecomputersociety.org/10.1109/LICS.1997.614927`.

[10] Roberto Di Cosmo, Delia Kesner & Emmanuel Polonovski (2003): *Proof Nets And Explicit Substitutions*. *Math. Str. in Comput. Sci.* 13(3), pp. 409–450. Available at http://dx.doi.org/10.1017/S0960129502003791.

[11] Maribel Fernández & Ian Mackie (2002): *Call-by-Value lambda-Graph Rewriting Without Rewriting*. In: *ICGT*, pp. 75–89. Available at http://dx.doi.org/10.1007/3-540-45832-8_8.

[12] Maribel Fernández & Nikolaos Siafakas (2009): *Labelled Lambda-calculi with Explicit Copy and Erase*. In: *LINEARITY*, pp. 49–64. Available at http://dx.doi.org/10.4204/EPTCS.22.5.

[13] Cormac Flanagan, Amr Sabry, Bruce F. Duba & Matthias Felleisen (1993): *The Essence of Compiling with Continuations*. In: *PLDI*, pp. 237–247. Available at http://doi.acm.org/10.1145/155090.155113.

[14] Jean-Yves Girard (1987): *Linear Logic*. *Theoretical Computer Science* 50, pp. 1–102. Available at http://dx.doi.org/10.1016/0304-3975(87)90045-4.

[15] Hugo Herbelin & Stéphane Zimmermann (2009): *An Operational Account of Call-by-Value Minimal and Classical lambda-Calculus in "Natural Deduction" Form*. In: *TLCA*, pp. 142–156. Available at http://dx.doi.org/10.1007/978-3-642-02273-9_12.

[16] Delia Kesner & Stéphane Lengrand (2007): *Resource operators for lambda-calculus*. *Inf. Comput.* 205(4), pp. 419–473. Available at http://dx.doi.org/10.1016/j.ic.2006.08.008.

[17] Olivier Laurent (1999): *Polarized Proof-Nets: Proof-Nets for LC*. In: *TLCA*, pp. 213–227. Available at http://dx.doi.org/10.1007/3-540-48959-2_16.

[18] Olivier Laurent (2002): *Étude de la polarisation en logique*. Thèse de doctorat, Université Aix-Marseille II.

[19] Olivier Laurent (2003): *Polarized proof-nets and λμ-calculus*. *Theor. Comput. Sci.* 290(1), pp. 161–188. Available at http://dx.doi.org/10.1016/S0304-3975(01)00297-3.

[20] Ian Mackie (2005): *Encoding Strategies in the Lambda Calculus with Interaction Nets*. In: *IFL*, pp. 19–36. Available at http://dx.doi.org/10.1007/11964681_2.

[21] John Maraist, Martin Odersky, David N. Turner & Philip Wadler (1999): *Call-by-name, Call-by-value, Call-by-need and the Linear lambda Calculus*. *Theor. Comput. Sci.* 228(1-2), pp. 175–210. Available at http://dx.doi.org/10.1016/S0304-3975(98)00358-2.

[22] Gordon D. Plotkin (1975): *Call-by-Name, Call-by-Value and the lambda-Calculus*. *Theor. Comput. Sci.* 1(2), pp. 125–159. Available at http://dx.doi.org/10.1016/0304-3975(75)90017-1.

[23] Alberto Pravato, Simona Ronchi Della Rocca & Luca Roversi (1999): *The call-by-value λ-calculus: a semantic investigation*. *Math. Str. in Comput. Sci.* 9(5), pp. 617–650. Available at http://dx.doi.org/10.1017/S0960129598002722.

[24] Laurent Regnier (1992): *Lambda-calcul et réseaux*. PhD thesis, Univ. Paris VII.

[25] Amr Sabry & Matthias Felleisen (1993): *Reasoning about Programs in Continuation-Passing Style*. *Lisp and Symbolic Computation* 6(3-4), pp. 289–360. Available at http://dx.doi.org/10.1007/BF01019462.

[26] Paolo Tranquilli (2009): *Nets Between Determinism and Nondeterminism*. Ph.D. thesis, Università degli Studi Roma Tre/Université Paris Diderot (Paris 7).

[27] Paolo Tranquilli (2011): *Intuitionistic differential nets and lambda-calculus*. *Theor. Comput. Sci.* 412(20), pp. 1979–1997. Available at http://dx.doi.org/10.1016/j.tcs.2010.12.022.

[28] Lionel Vaux (2007): *λ-calcul différentiel et logique classique: interactions calculatoires*. Ph.D. thesis, Université Aix-Marseille II.