Proof-nets and the bang calculus

Raffaele Di Donna* and Giulio Guerrieri†

May 29, 2025

Abstract

We study how the simply typed terms of the bang calculus can be translated into the proof nets of intuitionistic multiplicative and exponential linear logic (IMELL). We study the fragment of IMELL for the simply typed bang calculus by providing a sequentialization theorem characterizing all and only the proof nets corresponding to bang terms. We also show that cut-elimination simulates reduction, and that our translation factorizes the call-by-name and call-by-value translations of the simply typed lambda-calculus into IMELL passing through the simply typed bang calculus.

1 Introduction

Linear logic (*LL*), introduced by Girard in [4], is a refinement of classical and intuitionistic logic in which formulas are treated as resources: the structural rules of contraction and weakening are restricted to certain formulas, marked with modal operators, and the semantic equivalence induced by the models of *LL* is non-trivial. One of the main contributions of *LL* to the field of proof theory is the formalism of *proof-nets*: freed from the redundancy of sequent calculus concerning the order of the rules, it is a canonical notion of proof for multiplicative linear logic without the units, and allows cut elimination to be defined through local manipulations on a graph rather than global transformations on proof trees. It is therefore adequate to study the dynamics of normalization and to establish the most fundamental properties of the system, such as strong normalization (every reduction sequence is finite). The intuitionistic multiplicative exponential fragment *IMELL* of *LL* is expressive enough to contain minimal intuitionistic logic, and hence the λ-calculus, via two different translations defined by Girard [4], corresponding to different evaluation mechanisms, call-by-name [10, 7] and call-by-value [1].

In [2], a typed calculus was introduced to make a connection between *LL* and call-by-push-value [8, 9], a generalization of both call-by-name (cbn) and call-by-value (cbv) λ-calculi. A simplified version of this formalism is later developed in [3, 5] and dubbed "bang calculus" (bc), which captures both the cbn and cbv semantics of λ-calculus by internalizing Girard's translations in the calculus. The bc is essentially an extension of λ-calculus with an explicit ! constructor, which corresponds to the box construct of proof-nets and allows us to control precisely the evaluation strategy by distinguishing between functions and values.

We define a fragment !*IMELL* of *IMELL* containing the bc (simply typed according to Definition 2.2) and, in particular, the two Girard's translations (Proposition 2.5). We translate the bc into proof-nets of !*IMELL* satisfying simple geometric properties (Definitions 4.1 and 4.2) in such a way that Girard's cbn and cbv translations of λ-calculus into proof-nets can be factorized as the cbn and cbv embeddings of λ-calculus into bc, followed by the translation of bc into proof-nets (Remark 4.4). A sequentialization theorem holds for both the logical fragment (Theorem 3.7) and the bc (Theorem 4.6). Finally, the dynamics of bc is simulated by the cut elimination procedure modulo conversion of proof-structures (Definition 3.3 and Theorem 4.5).

*Université Paris Cité, Paris, France and Università Roma Tre, Rome, Italy, didonna@irif.fr
†University of Sussex, Brighton, UK g.guerrieri@sussex.ac.uk

2 IMELL and the simply typed bang calculus

The formulas of intuitionistic multiplicative exponential linear logic (*IMELL*) can be defined by the following grammar, given a countably infinite set of atomic formulas denoted by $X, Y, Z, \dots$:

$$O ::= X \mid O \otimes O \mid O \parr I \mid I \parr O \mid !O \quad (\textit{output}) \qquad I ::= X^\perp \mid I \parr I \mid I \otimes O \mid O \otimes I \mid ?I \quad (\textit{input})$$

The fragment !*IMELL* of *IMELL* is given by the grammar:

$$O ::= X \mid ?I \parr O \mid !O \quad (\textit{output}) \qquad I ::= X^\perp \mid !O \otimes I \mid ?I \quad (\textit{input})$$

We set $O \multimap O' := O^\perp \parr O'$, where the involutive *negation* $A^\perp$ of a (input or output) formula A is defined by de Morgan's laws $(A \otimes B)^\perp = A^\perp \parr B^\perp$, $(!A)^\perp = ?A^\perp$ and $A^{\perp\perp} = A$. Note that the negation of an input formula of !*IMELL* (resp. *IMELL*) is an output formula of !*IMELL* (resp. *IMELL*) and vice-versa.

The *IMELL* sequent calculus is the one-sided *LL* sequent calculus restricted to sequents of *IMELL* formulas. It is easy to check that every provable sequent of *IMELL* formulas contains exactly one output formula.

Definition 2.1. *Bang terms* are defined by the grammar $t ::= x \mid \lambda x\, t \mid \langle t\rangle t \mid t^!$, where x denotes any variable. We define $\operatorname{der} t := \langle \lambda x\, x\rangle t$. We obtain a rewriting system called *bang calculus* (*bc*) by defining the root rewriting step $\langle \lambda x\, t\rangle s^! \mapsto_{\mathsf{b}} t\{s/x\}$ and by extending it in the usual inductive way to a reduction step $\to_{\mathsf{b}}$.

The (Curry-style) *simply typed bang calculus* endows bang terms with output formulas of !*IMELL* according to the rules defined below. The intuition for is that terms in bc are either functions ($!O \multimap O'$) or values ($!O$).

Definition 2.2. Typing derivations of bang terms (where $\Gamma = x_1 : !O_1, \dots, x_k : !O_k$ for some $k \geq 0$):

$$\pi^x_\Gamma : \frac{}{\Gamma, x : !O \vdash x : O}\, var \qquad \lambda x\, \pi : \frac{\vdots\ \pi}{\Gamma, x : !O \vdash t : O'}\, \lambda \qquad \langle \pi_t \rangle \pi_s : \frac{\vdots\ \pi_t \quad \vdots\ \pi_s}{\Gamma \vdash t : !O \multimap O' \qquad \Gamma \vdash s : !O}\, app \qquad \pi^! : \frac{\vdots\ \pi}{\Gamma \vdash t : O}\, !$$

Definition 2.3. The translations $(\cdot)^{\mathsf{n}}$ and $(\cdot)^{\mathsf{v}}$ of λ-terms into bc are defined by induction on λ-terms:

$$x^{\mathsf{n}} := x \qquad (\lambda x M)^{\mathsf{n}} := \lambda x M^{\mathsf{n}} \qquad ((M)N)^{\mathsf{n}} := \langle M^{\mathsf{n}}\rangle (N^{\mathsf{n}})^!$$
$$x^{\mathsf{v}} := x^! \qquad (\lambda x M)^{\mathsf{v}} := (\lambda x M^{\mathsf{v}})^! \qquad ((M)N)^{\mathsf{v}} := \langle \operatorname{der} M^{\mathsf{v}}\rangle N^{\mathsf{v}}$$

Simple types are defined by the grammar $T ::= \alpha \mid T \Rightarrow T$. The translations $(\cdot)^{\mathsf{N}}$ and $(\cdot)^{\mathsf{V}}$ of simple types into *IMELL* are defined by induction on simple types:

$$X^{\mathsf{N}} := X \qquad (T \Rightarrow S)^{\mathsf{N}} := !T^{\mathsf{N}} \multimap S^{\mathsf{N}} \qquad X^{\mathsf{V}} := X \qquad (T \Rightarrow S)^{\mathsf{V}} := !T^{\mathsf{V}} \multimap !S^{\mathsf{V}}$$

A *simple* (resp. *bang*) *context* is a finite partial function from variables to simple types (resp. *IMELL* formulas). We write $\Gamma = x_1 : A_1, \dots, x_n : A_n$ when the domain of the context Γ is $\mathsf{dom}(\Gamma) = \{x_1, \dots, x_n\}$ and $\Gamma(x_i) = A_i$ for every $i \in \{1, \dots, n\}$. If $\Gamma = x_1 : T_1, \dots, x_k : T_k$ is a simple context, the bang contexts $x_1 : !T_1^{\mathsf{N}}, \dots, x_k : !T_k^{\mathsf{N}}$ and $x_1 : !T_1^{\mathsf{V}}, \dots, x_k : !T_k^{\mathsf{V}}$ are denoted by $!\Gamma^{\mathsf{N}}$ and $!\Gamma^{\mathsf{V}}$ respectively.

Remark 2.4. For every simple type T, the formulas T^{N} and T^{V} are output in !*IMELL*.

Proposition 2.5. *Let M be a λ-term. If $\Gamma \vdash M : T$, then $!\Gamma^{\mathsf{N}} \vdash M^{\mathsf{n}} : T^{\mathsf{N}}$ and $!\Gamma^{\mathsf{V}} \vdash M^{\mathsf{v}} : !T^{\mathsf{V}}$.*

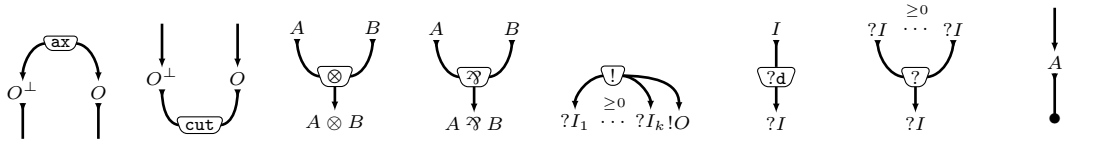
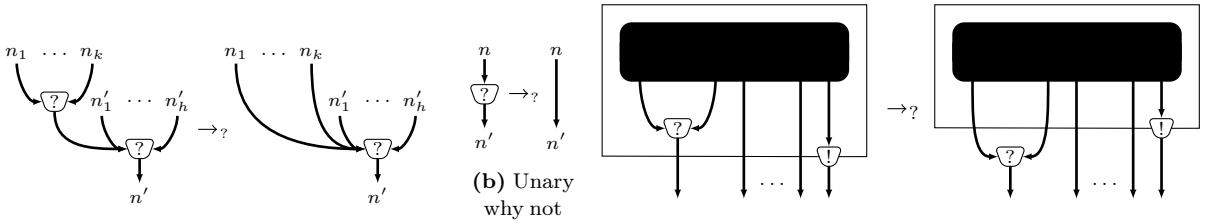


Figure 1: The local constraints that a node labeled by ℓ has to respect.



(a) Two why not nodes on top of each other.

(b) Unary why not node.

(c) Why not nodes and box borders.

Figure 2: Conversion steps are defined locally, by erasing nodes, moving them across box borders, and re-routing arcs.

3 Proof-nets for IMELL

Definition 3.1. A *ground structure* is a finite acyclic directed graph where its arcs are labeled by *IMELL* formulas (called *types*), its vertices (or *nodes*) are labeled by exactly one symbol between `ax`, `cut`, $\otimes$, $\parr$, !, ?d, ?, $\bullet$, and such that every node satisfies the local constraints illustrated in Figure 1.

Let G be a ground structure and let n be a node of G. A *premise* (resp. *conclusion*) of n is an arc incoming (resp. outgoing) n. If n is a `cut` node, then n is an *axiom cut* if at least one premise of n is the conclusion of an `ax` node, a *multiplicative cut* if its premises are conclusions of a $\otimes$ node and of a $\parr$ node. The premises of the $\bullet$ nodes of G are called the *conclusions of* G. A node of G is *terminal* if all its conclusions are conclusions of G.

Definition 3.2. An *IMELL*$_0$ *proof-structure* is a ground structure without ! nodes. An *IMELL*$_{d+1}$ *proof-structure* is a ground structure R together with a function f that associates a *IMELL*$_d$ proof-structure with conclusions of type $?I_1, \dots, ?I_k, O$ with every ! node n of R with conclusions of type $?I_1, \dots, ?I_k, O$. We write $R' \sqsubset R$ when $R' = f(n)$ for some ! node n of R. The *unfolding* of R, denoted by $|R|$, is the graph obtained from R by replacing every ! node n of R with conclusions of type $?I_1, \dots, ?I_k, !O$ with $|f(n)|$ where a unary ! node has been added under its unique output conclusion. A(n *IMELL*) *proof-structure* is a *IMELL*$_d$ proof-structure for some $d \geq 0$.

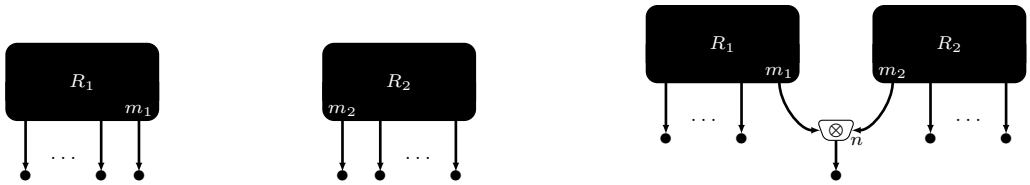
Intuitively, the function f associates with each ! node the content of its box. We often leave f implicit.

The set of proof-structures is endowed with two reduction relations: cut elimination steps, which are the usual ones for multiplicative exponential linear logic, and the conversion steps given by the following definition.

Definition 3.3. Conversion steps on proof-structures are defined in Figure 2. For all proof-structures R and R', we write $R \to R'$ (resp. $R \to_? R'$) if R reduces to R' in exactly one cut elimination (resp. conversion) step. The smallest equivalence relation containing $\to_?$ is called *conversion* and is denoted by $\simeq_?$.

Proof-structures corresponding to correct proofs can be discerned via the notion of switching, as we will see.

Definition 3.4. A *switching* of a proof-structure R is a function φ mapping every $\parr$ or ? node with at least two premises to one of its premises. The *switching graph* of R induced by φ, denoted by R^φ, is the graph obtained from R by replacing the target of the arc a by a fresh $\bullet$ node, for every premise a of a $\parr$ or ? node n such that $\varphi(n) \neq a$. We say that R is a *proof-net* if R has exactly one output conclusion, the underlying undirected graph of R^φ is acyclic and every $R' \sqsubset R$ is a proof-net.



(a) A proof-structure R_1 having m_1 among its terminal nodes.

(b) A proof-structure R_2 having m_2 among its terminal nodes.

(c) The proof-structure R is obtained by taking the disjoint union of R_1 and R_2 and by adding a terminal $\otimes$ node n under m_1 and m_2.

Figure 3: The proof-structures R_1, R_2 and R in the hypotheses of Lemma 3.6.

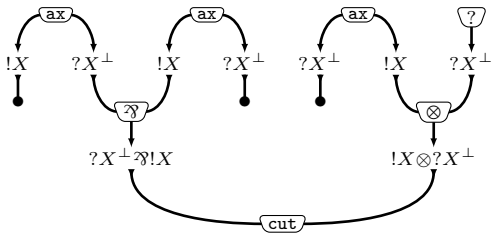


Figure 4: A proof-net of !*MELL* which is not sequentializable in !*MELL* and has a multiplicative cut.

Cut elimination is confluent and, when restricted to axiom and multiplicative steps, strongly normalizing.

Every sequent calculus proof π in *IMELL* with conclusion the sequent $I_1, \dots, I_k, O$ can be desequantialized, that is, translated into a proof-structure π^* with conclusions of type $I_1, \dots, I_k, O$. What about the converse?

Definition 3.5. Let $\mathcal{F}$ be a fragment of *IMELL*. A proof-structure R of $\mathcal{F}$ is *sequentializable in* $\mathcal{F}$ if there exists a sequent calculus proof π in $\mathcal{F}$ such that $R = \pi^*$.

The key result allowing us to prove our sequentialization results is the following lemma.

Lemma 3.6. *Let R_1, R_2 and R be the proof-structures depicted in Figure 3. If R_1 and R_2 are sequentializable in IMELL, and if the last rule of every sequent calculus proof π in IMELL such that $R = \pi^*$ corresponds to n or to a terminal node of R_2 (resp. R_1), then the last rule of any sequent calculus proof π_0 in IMELL such that $R_1 = \pi_0^*$ (resp. $R_2 = \pi_0^*$) corresponds to m_1 (resp. m_2).*

Theorem 3.7 (Sequentialization in IMELL without multiplicative cuts)**.** *Let R be a IMELLproof-structure such that $|R|$ has no multiplicative cuts: R is a proof-net if and only if R is sequentializable in IMELL.*

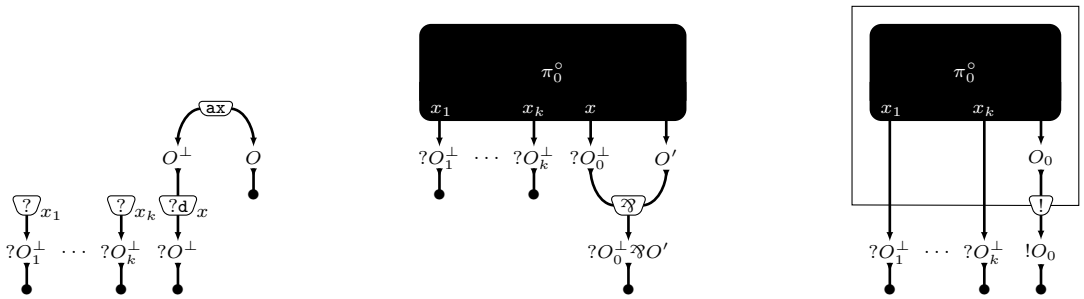
Remark 3.8. The previous result cannot be extended to all proof-nets of *IMELL* (see the example in Figure 4).

We stress that Theorem 3.7 is an original contribution, different from the characterization via Lamarche's essential nets [6], which relies on a notion of jump, while we do not. The trade-off is that we must assume the absence of multiplicative cuts and this hypothesis is essential, as Remark 3.8 shows even for the fragment !*IMELL*.

4 Translations

We can now translate typing derivations of bang terms into proof-structures via two translations, and characterize among the proof-structures the ones that correspond to typing derivations of bang terms.

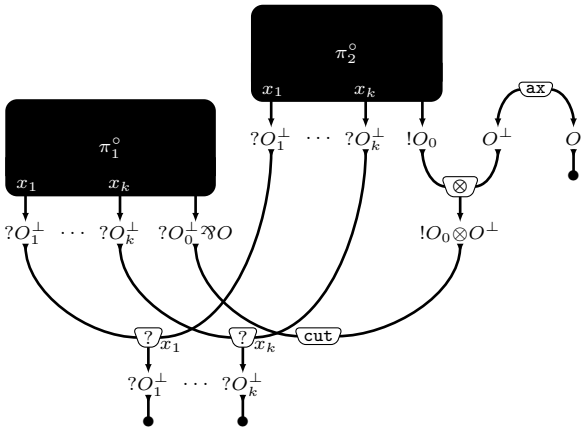
Definition 4.1. A proof-structure R is a *bang proof-structure* if it is labeled by !*IMELL* formulas, and every input premise of a $\parr$ or $\bullet$ node of R is the conclusion of a ?d or ? node and every $R' \sqsubset R$ is a bang proof-structure.



(a) *var*: $\Gamma = x_1 : !O_1, \dots, x_k : !O_k, x : !O$ and $t = x$.

(b) λ: $\Gamma = x_1 : !O_1, \dots, x_k : !O_k$ and $\pi = \lambda x \pi_0$. By induction hypothesis on π_0, we know that π_0° is defined.

(c) λ: $\Gamma = x_1 : !O_1, \dots, x_k : !O_k$ and $\pi = \pi_0^!$. By induction hypothesis on π_0, we know that π_0° is defined.



(d) *app*: $\Gamma = x_1 : !O_1, \dots, x_k : !O_k$ and $\pi = \langle \pi_1 \rangle \pi_2$. By induction hypothesis on π_1 and on π_2, we know that π_1° and π_2° are defined.

Figure 5: The translation π° of a typing derivation π of $\Gamma \vdash t : O$, defined by induction on π.

A bang proof-structure is a *bang proof-net* if it is also a proof-net. A bang proof-structure R is *named* if it comes with a function ν which associates a variable with each ?d or ? node of $|R|$, in such a way that, for every ? node n whose premises are conclusions of $n_1, \dots, n_k$, we have $\nu(n) = \nu(n_i)$ for all $i \in \{1, \dots, k\}$.

Definition 4.2. Let t be a bang term and let π be a typing derivation of $\Gamma \vdash t : O$. We define the named bang proof-structure π° of !*IMELL* by induction on π, as depicted in Figure 5. We define $\pi^\bullet$ as the named bang proof-structure of !*IMELL* obtained from π° by normalizing the axiom and multiplicative cut elimination steps.

Remark 4.3. The named bang proof-structures π° and $\pi^\bullet$ of !*MELL* are actually named bang proof-nets.

Remark 4.4. The translations of cbn [10, 7] and cbv [1] λ-calculi into proof-nets can be factorized as the cbn and cbv embeddings of λ-calculus into bc, followed by the translation of bc into named bang proof-nets.

Every reduction step in bc is simulated by a finite number of cut elimination steps, modulo conversion.

Theorem 4.5 (Simulation)**.** *Let t and s be bang terms and let π_t and π_s be typing derivations of $\Gamma \vdash t : O$ and of $\Gamma \vdash s : O$ respectively. If $t \to_{\mathtt{b}} s$, there exist bang proof-nets R_t, R_s such that $\pi_t^\circ \simeq_? R_t$, $\pi_s^\circ \simeq_? R_s$ and $R_t \to^* R_s$.*

Finally, named bang proof-nets with no axiom or multiplicative cuts are always translations of bang terms.

Theorem 4.6 (Sequentialization)**.** *Let R be a named bang proof-net with conclusions $c_0, c_1, \dots c_k$, where c_0 has an output type O and, for all $i \in \{1, \dots, k\}$, the conclusion c_i has an input type $?O_i^\perp$ and is labeled by the variable*

x_i. *If* $|R|$ *has no axiom cuts and no multiplicative cuts, then there exist a bang term* t *and a typing derivation* π *of* $x_1 : !O_1, \dots, x_k : !O_k \vdash t : O$ *such that* $\pi^\bullet = R$.

References

[1] Beniamino Accattoli. Proof nets and the call-by-value λ-calculus. *Theor. Comput. Sci.*, 606:2–24, 2015.

[2] Thomas Ehrhard. Call-by-push-value from a linear logic point of view. In *Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Proceedings*, volume 9632 of *Lecture Notes in Computer Science*, pages 202–228. Springer, 2016.

[3] Thomas Ehrhard and Giulio Guerrieri. The bang calculus: an untyped lambda-calculus generalizing call-by-name and call-by-value. In James Cheney and Germán Vidal, editors, *Proceedings of the 18th International Symposium on Principles and Practice of Declarative Programming, Edinburgh, United Kingdom, September 5-7, 2016*, pages 174–187. ACM, 2016.

[4] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[5] Giulio Guerrieri and Giulio Manzonetto. The bang calculus and the two girard's translations. In *Proceedings Joint International Workshop on Linearity & Trends in Linear Logic and Applications, Linearity-TLLA@FLoC 2018*, volume 292 of *EPTCS*, pages 15–30, 2018.

[6] François Lamarche. Proof Nets for Intuitionistic Linear Logic: Essential Nets. Research report, 2008.

[7] Olivier Laurent. Polarized proof-nets and $\lambda\mu$-calculus. *Theoretical Computer Science*, 290(1):161–188, 2003.

[8] Paul Blain Levy. *Call-By-Push-Value: A Functional/Imperative Synthesis*, volume 2 of *Semantics Structures in Computation*. Springer, 2004.

[9] Paul Blain Levy. Call-by-push-value: Decomposing call-by-value and call-by-name. *High. Order Symb. Comput.*, 19(4):377–414, 2006.

[10] Laurent Regnier. Lambda-calcul et réseaux. *Thèse de doctorat, Université Paris VII*, 1992.