About Translations of Classical Logic into Polarized Linear Logic

Olivier Laurent
Preuves Programmes Systèmes
CNRS - Paris 7
Olivier.Laurent@pps.jussieu.fr

Laurent Regnier
Institut de Mathématiques de Luminy
CNRS
regnier@iml.univ-mrs.fr

Abstract

We show that the decomposition of Intuitionistic Logic into Linear Logic along the equation $A \to B = !A \multimap B$ may be adapted into a decomposition of classical logic into LLP*, the* polarized *version of Linear Logic. Firstly we build a categorical model of classical logic (a Control Category) from a categorical model of Linear Logic by a construction similar to the co-Kleisli category. Secondly we analyse two standard Continuation-Passing Style (CPS) translations, the Plotkin and the Krivine's translations, which are shown to correspond to two embeddings of* LLP *into* LL.

1. Introduction

Curry-Howard for classical logic. Thanks to the seminal work of Griffin [10] who showed that classical principles could be used to type control operators such as Scheme's `call/cc`, it is now well known that classical logic (LK), as intuitionistic logic (LJ), fits into the Curry-Howard paradigm. Work on *CPS*-translations showing that they correspond to $\neg\neg$-translations [21] à la Gödel, have been used to compile these control operators into lambda-calculus. From the viewpoint of logic, work has been carried out on the so-called "classical constructivization" problem, and proof systems have been designed for classical logic enjoying properties analogous to those of intuitionistic logic: normalization, Church-Rosser, denotational semantics… Let us mention Girard's system LC which introduces the important notion of *polarities* in logic [9], Parigot's lambda-mu-calculus [22] which is a natural extension of lambda-calculus embodying classical principles, and Danos-Joinet-Schellinx (DJS)'s impressive classification of classical constructivizations within the syntactical framework $\mathsf{LK}^{\mathsf{tq}}$ [6, 7].

Decomposing the intuitionistic implication. An important advance in the study of (typed) lambda-calculus, was the discovery by Girard [8] that the intuitionistic implication could be decomposed in linear logic (LL) along the now standard equation

$$A \to B = !A \multimap B$$

yielding the so-called Girard's translation: $\mathsf{LJ} \xrightarrow{!A \multimap B} \mathsf{LL}$. This decomposition has also a categorical version by means of the construction of a co-Kleisli of a monoidal closed category.

Classical logic and linear logic. In this paper we will define some translations of classical logic into linear logic. We use lambda-mu calculus (that is, classical natural deduction) as our classical proof system. When dealing with lambda-mu calculus as a logical system we shall spell it LK, which shouldn't be confused with Gentzen's sequent calculus as the latter is not constructive. We stick to the LK notation to enforce the idea that our study does not depend on the chosen system and could be carried out with any *constructive* classical proof-system.

Such translations are easy to construct simply by composing a CPS translation with a translation of LJ into LL, e.g. Girard's translation:

$$\mathsf{LK} \xrightarrow{\mathsf{CPS}} \mathsf{LJ} \xrightarrow{!A \multimap B} \mathsf{LL}$$

For example if we consider the call by name (CBN) Plotkin's translation [23], this yields $\mathsf{LK} \xrightarrow{!?!A \multimap ?!B} \mathsf{LL}$ where the formula on the arrow is the linear formula associated to the implication of LK. As we see the defect of this method is that it makes intensive use of exponentials hiding the underlying linear structure.

As a by-product of their work on $\mathsf{LK}^{\mathsf{tq}}$, DJS have ameliorated this translation producing $\mathsf{LK} \xrightarrow{!?A \multimap ?B} \mathsf{LL}$ which makes a more sensible use of exponentials. This amelioration relies on the remark that, in a CBN setting, continuations are linear. The intuitive reason is that a continuation is essentially a functional applying its function argument to a stack of arguments; it is well known that this operation is linear in the function argument.

We shall take advantage of the linearity of continuations at the level of the target language; so we will use a target language allowing the expression of linearity constraints. More precisely, the CLC calculus is defined to be regular lambda-calculus typed with usual types and linear types; this kind of calculus has already been used in work analyzing linearity in CPS translations, e.g. by Berdine et al. [2] and Laird [15]. It allows us to give a clear factorization of DJS translation:

$$\begin{tikzcd} \mathsf{LK} \arrow[r, "\mathsf{CPS}"] \arrow[dr, "!?A\multimap ?B"'] & \mathsf{CLC} \arrow[d, "!A\multimap B"] \\ & \mathsf{LL} \end{tikzcd}$$

Linearity of continuations vs. continuation-passing. The linearity of CBN continuations should be opposed to the linearity of *continuation-passing* which occurs in CPS translations of the CBV lambda-calculus. This fact has been remarked and analysed from a syntactical viewpoint by the above mentioned work of Berdine et al., and more recently, by Laird who uses a semantical approach; both prove a full abstraction result for the translation. However as both authors point, linearity of continuation-passing is only true in the intuitionistic setting (without `call/cc`), whereas linearity of continuations remains true even when switching to the classical and CBN world, as we do here.

Introducing polarities. An important remark is that the target language of the translation depicted in the above diagram actually lies inside LL_{pol}, the fragment of LL consisting of well polarized formulas (in the sense of Girard). In fact all the classical and intuitionistic types are translated by negative formulas. The first author has introduced a proof system LLP (Polarized Linear Logic) for polarized formulas, which relaxes the use of structural rules on negative formulas [16]. This system is better suited than LL or LL_{pol} for encoding classical logic because it allows to design a translation $\mathsf{LK} \xrightarrow{!A\multimap B} \mathsf{LLP}$ which is a straightforward extension of Girard's translation.

The main achievement of this paper is to give a precise correspondence between CPS translations of LK into LJ and translations of LLP into LL (really LL_{pol}). We study Plotkin's and Krivine's translations and show for each of them that we get the following diagram:

$$\begin{tikzcd} \mathsf{LK} \arrow[r, "\mathsf{CPS}"] \arrow[d, "!A\multimap B"'] & \mathsf{CLC} \arrow[d, "!A\multimap B"] \\ \mathsf{LLP} \arrow[r] & \mathsf{LL} \end{tikzcd}$$

where the bottom horizontal arrow represents respectively the so-called *box* and *reversing* translations. This shows that LLP stands in the same relation with respect to LL than classical logic with respect to intuitionistic logic, thus giving a notion of CPS translation internal to linear logic which can be used to compare them as we do with Krivine's and Plotkin's translations. Typically the same work could be achieved with the Hofmann-Streicher translation [11] which is identified with Plotkin's in the linear world.

Categorical semantics. The paper begins with the construction of a categorical model of classical logic, a control category as defined by Selinger [25], by a method very similar to the construction of the co-Kleisli of a monoidal closed category. This construction may be seen as a semantical version of the linear analysis of classical logic by LLP.

Logical systems used. We will take the Parigot's call by name lambda-mu-calculus [22] as source system for the CPS-translations. This calculus is probably the most suitable for our purpose since it naturally features the Curry-Howard correspondence for classical logic: from the logical point of view it is a natural deduction system for classical logic and from the computer science point of view it is an extension of lambda-calculus encoding `call/cc` like operators.

Our target language for CPS translations will be regular lambda-calculus. We however change the typing assignment system to allow the expression of linearity constraints on continuations, getting a system called CLC which, as pointed above, is very similar to Berdine et al. and/or Laird's target language, and may also be seen as a subsystem of Barber-Plotkin's DILL [1].

The call by value case. For the sake of simplicity we stick to CBN calculus and translations. Let us point though that the call by value (CBV) case is symmetrical: instead of Girard's translation, use the CBV translation $\mathsf{LK} \xrightarrow{!(A\multimap ?B)} \mathsf{LLP}$ which associates *positive* formulas to classical and intuitionistic types; then one may show that Plotkin's CBV translation is associated with the same box translation of LLP into LL [16].

Simulation properties. The paper will define various translations between logical systems. Let us claim once for all that all of them are *reduction preserving*, that is for each translation $t \mapsto |t|$ the following diagram commutes:

$$\begin{tikzcd} t \arrow[r] \arrow[d] & {|t|} \arrow[d] \\ t' \arrow[r] & {|t|' = |t'|} \end{tikzcd}$$

where the horizontal arrows are translations and the vertical ones are reductions.

2. Polarized linear systems

We derive two proof systems from Linear Logic based on the introduction of *polarities* in linear formulas: *negative*

(N) and *positive* (P) formulas are defined by:

$$\begin{array}{rcl} N & ::= & X \mid \bot \mid N \parr N \mid \top \mid N \,\&\, N \mid ?P \;[\; \mid ?N \;] \\ P & ::= & X^\bot \mid 1 \mid P \otimes P \mid 0 \mid P \oplus P \mid !N \;[\; \mid !P \;] \end{array}$$

Formulas of the form $?N$ (resp. $!P$), that is the bracketed cases above, will be said *weakly* negative (resp. weakly positive).

The system LL_{pol} is simply the restriction of linear logic to polarized formulas (including weakly polarized ones); thus it is a fragment of Linear Logic in the natural sense. As a side-effect of this restriction, one easily checks that all the rules preserve the fact that at most one positive formula may occur in a polarized sequent.

The second system, named LLP, is further restricted to strongly polarized formulas only but also adds structural rules for any negative formula (whereas Linear Logic has structural rules only for $?$ formulas):

$$\frac{\vdash \Gamma, N, N}{\vdash \Gamma, N} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, N} \qquad \frac{\vdash \mathcal{N}, N}{\vdash \mathcal{N}, !N}$$

where $\mathcal{N}$ is a context of negative formulas; again one easily sees that at most one positive formula occurs in any LLP sequent. The generalization of LL structural rules, apparently innocuous, has rather deep consequences and LLP may be considered as a new system related to, but independent from LL (see [16] for an extensive study of LLP).

The study of LLP has lead to two translations of LLP into LL_{pol} that we are going to describe now. For the sake of simplicity, as these translations are meant to be related with CPS translations of lambda-mu calculus, we will only consider lambda-mu calculus without products and LLP without additives. The work presented is however easily extendible to the more general setting.

The box translation. The first translation is based on a decoration of LLP derivations by exponential connectives in the spirit of decorations of classical logic into linear logic by Danos-Joinet-Schellinx [7]; it has been used to prove the strong normalization of LLP in [18]. Viewed from the positive side, this translation encapsulates every positive formula of a proof into a $!$-box, thus its name.

The goal is to make LL-correct each application of a generalized structural rule by decorating it with the adequate exponential. To achieve this, as any negative formula may be the premisse of a structural rule, they are decorated with a $?$ in the following way:

$$\begin{aligned} X^b &= ?X \\ (N \parr M)^b &= ?(N^b \parr M^b) \\ (?P)^b &= ?P^b \end{aligned}$$

and dually for the positive formulas. Sequents $\vdash \Gamma$ are translated as $\vdash \Gamma^b$.

The decoration of proofs just requires to add dereliction and promotion rules corresponding to the exponential decoration of formulas. We show only the translation of the tensor and contraction rules:

$$\dfrac{\dfrac{\dfrac{\vdash \Gamma^b, P^b \qquad \vdash \Delta^b, Q^b}{\vdash \Gamma^b, \Delta^b, P^b \otimes Q^b}}{\vdash \Gamma^b, \Delta^b, !(P^b \otimes Q^b)}}{\vdash \Gamma^b, \Delta^b, (P \otimes Q)^b}$$

and

$$\dfrac{\dfrac{\dfrac{\vdash \Gamma^b, (N \parr M)^b, (N \parr M)^b}{\vdash \Gamma^b, ?(N^b \parr M^b), ?(N^b \parr M^b)}}{\vdash \Gamma^b, ?(N^b \parr M^b)}}{\vdash \Gamma^b, (N \parr M)^b}$$

where the dashed lines indicate equality between sequents; note that the promotion rule in the translation of the $\otimes$ is LL correct because, P and Q being positive, Γ and Δ contain only negative formulas the translations of which begin with a $?$.

The reversing translation. The second translation, that has been used by Tortora de Falco and Quatrini [24] to refine the DJS embedding of LK into LL, uses a lighter decoration:

$$\begin{aligned} X^\rho &= ?X^\bot \\ (N \parr M)^\rho &= N^\rho \parr M^\rho \\ (?P)^\rho &= ?P^\rho \end{aligned}$$

and dually for the positive formulas. Sequents $\vdash \Gamma$ are translated as $\vdash \Gamma^\rho$. Note that, contrasting with the box translation, the reversing translation of a sequent is strongly polarized.

We still have to transform any negative formula premisse of a structural rule into a $?$ formula to be able to apply the rule in LL_{pol} but this time we will use a syntactical version of the $?$-algebra morphisms (see below), that is proofs ρ_N of $\vdash !N^{\rho\bot}, N^\rho$ for any negative formula N. One easily shows by induction on N that such a proof exists (this fact was used in particular by DJS who called this class of proofs the *$?$-fix proofs*; they were meant to achieve the so-called *reversing* of proofs). We show only the translation of a contraction rule, the other rules being treated similarly:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\vdash \Gamma^\rho, N^\rho, N^\rho}}{\vdash \Gamma^\rho, ?N^\rho, N^\rho}}{\vdash \Gamma^\rho, ?N^\rho, ?N^\rho}}{\vdash \Gamma^\rho, ?N^\rho} \qquad \dfrac{\rho_N}{\vdash !N^{\rho\bot}, N^\rho}$$
$$\overline{\qquad\qquad \vdash \Gamma^\rho, N^\rho \qquad\qquad}$$

Note that contrarily to the box translation, this one introduces cut rules. For this reason in order to plainly fulfill our claim that it is reduction preserving one should really define the translation of a proof as the one obtained after reducing these cuts.

3. A construction of a control category

In this section we will give a categorical counterpart of the syntactical translation of LLP into $\mathsf{LL}_{\mathsf{pol}}$ by extracting a control category (thus a model of LLP, see [16]) from a categorical model of $\mathsf{LL}_{\mathsf{pol}}$ (of full LL really).

3.1. Notations and terminology

We first briefly record known facts relative to the categorical semantics of linear logic and set up the notations used throughout.

The $*$-autonomous structure. Let $\mathcal{C}$ be a model of MALL (multiplicative, additive linear logic), that is a $*$-autonomous category with finite products (and finite coproducts by duality). We follow the linear logic notations and denote: 1 the unit of the tensor, $\perp$ the dualizing object (thus dual of 1), $⅋$ the cotensor (by which we mean that $⅋$ is defined as the dual of $\otimes$ but this shouldn't hide the fact that $⅋$ really is also a tensor product) so that $\perp$ is the neutral of $⅋$, & the cartesian product, $\oplus$ the sum (dual of &), 0 and $\top$ the units of $\oplus$ and &.

The terminology is slightly unfortunate as we will mainly define our constructions with respect to the cotensor $⅋$ which therefore would be better named "tensor". This choice is reflecting the global choice of the paper to treat CBN calculus which is naturally associated to the *negative* fragment of LLP.

If $f : A \to B$ is a $\mathcal{C}$-morphism then $f^\perp : B^\perp \to A^\perp$ is the dual morphism. Finally we denote $\varepsilon_{A,B} : (A ⅋ B^\perp) \otimes B \to A$ the evaluation morphism in $\mathcal{C}$; it is natural in A and also satisfies a kind of naturality in B.

Transposing a morphism. The $*$-autonomous structure yields a bijection between $\mathcal{C}(A, B ⅋ C)$ and $\mathcal{C}(A \otimes C^\perp, B)$ which associates to any $f : A \otimes C^\perp \to B$ a unique $\tilde{f} : A \to B ⅋ C$ called the *transposed* of f such that

$$f = A \otimes C^\perp \xrightarrow{\tilde{f} \otimes C^\perp} (B ⅋ C) \otimes C^\perp \xrightarrow{\varepsilon_{B,C^\perp}} B$$

$⅋$-monoids. A *$⅋$-monoid* on $\mathcal{C}$ is a triple $N = (\bar{N}, c_N, w_N)$ where $\bar{N}$ is an object of $\mathcal{C}$, $c_N : \bar{N} ⅋ \bar{N} \to \bar{N}$ and $w_N : \perp \to \bar{N}$ are morphisms of $\mathcal{C}$ satisfying the standard neutral, associativity an commutativity equations. $⅋$-monoids are meant to interpret negative formulas of LLP.

A *$\otimes$-comonoid* P is the dual of a $⅋$-monoid, that is, a triple $(\bar{P}, c_P, w_P)$ where $c_P : \bar{P} \to \bar{P} \otimes \bar{P}$ and $w_P : \bar{P} \to 1$ are such that $(\bar{P}^\perp, c_P{}^\perp, w_P{}^\perp)$ is a $⅋$-monoid.

Given two $⅋$-monoids M and N, a *$⅋$-morphism* is a monoidal $\mathcal{C}$-morphism $f : \bar{M} \to \bar{N}$, that is, f commutes with the $⅋$-monoid structure. We denote by $⅋\text{-}\mathsf{Mon}(\mathcal{C})$ the category of $⅋$-monoids and $⅋$-morphisms. Given two $⅋$-monoids M and N, and using the associativity of $⅋$, one may build morphisms $c_{M⅋N} : (\bar{M} ⅋ \bar{N}) ⅋ (\bar{M} ⅋ \bar{N}) \to \bar{M} ⅋ \bar{N}$ and $w_{M⅋N} : \perp \to \bar{M} ⅋ \bar{N}$ giving $M ⅋ N$ a $⅋$-monoid structure. Clearly $⅋$ is a tensor product on $⅋\text{-}\mathsf{Mon}(\mathcal{C})$. Interestingly it is also a coproduct. One may also show that & is a cartesian product on $⅋\text{-}\mathsf{Mon}(\mathcal{C})$.

Adjunction. Let $U_\mathcal{C} : ⅋\text{-}\mathsf{Mon}(\mathcal{C}) \to \mathcal{C}$ be the forgetful functor. In the sequel we will suppose that $\mathcal{C}$ has the property that *$U_\mathcal{C}$ has a left adjoint* $?_\mathcal{C}$ [1]. A typical example of such a situation is given by the category of coherent spaces with the multiset version of !.

We denote by ? the endofunctor $U_\mathcal{C} \circ ?_\mathcal{C}$ on $\mathcal{C}$ (and by ! its dual). It is standard fact that it is a monad on $\mathcal{C}$ (see [19]). Still in accordance with linear logic terminology we will denote $\mathsf{der}_A : A \to ?A$ and $\mathsf{dig}_A : ??A \to ?A$ the natural transformations associated to ?. To keep notations light we will also denote $\mathsf{der}_A : !A \to A$ and $\mathsf{dig}_A : !A \to !!A$ the dual natural transformations associated to the comonad !.

The property that $U_\mathcal{C}$ has a left adjoint makes $\mathcal{C}$ a *Lafont category* [14]. A detailed proof that Lafont's categories are sound models of linear logic has been produced by Bierman (see [3, 4]). It uses some interesting consequences of the existence of an adjunction among which:

- for any pair of objects A and B we have $?(A \oplus B) \doteq ?A ⅋ ?B$.
- Any $⅋$-monoid N is a ?-algebra and any $⅋$-morphism is a ?-algebra morphism, that is, there is a $⅋$-morphism $\mathsf{alg}_N : ?\bar{N} \to \bar{N}$, natural in N and satisfying:

  $$\bar{N} \xrightarrow{\mathsf{der}_{\bar{N}}} ?\bar{N} \xrightarrow{\mathsf{alg}_N} \bar{N} = \mathsf{Id}_{\bar{N}}$$
  $$??\bar{N} \xrightarrow{\mathsf{dig}_{\bar{N}}} ?\bar{N} \xrightarrow{\mathsf{alg}_N} \bar{N} = ??\bar{N} \xrightarrow{?\mathsf{alg}_N} ?\bar{N} \xrightarrow{\mathsf{alg}_N} \bar{N}$$

  such that $⅋$-morphisms commute with the ?-algebra structure.

  Dually any $\otimes$-comonoid P has a !-coalgebra structure $\mathsf{coalg}_P : \bar{P} \to !\bar{P}$.
- The monad ? is *monoidal*, that is, for each pair of objects A and B there is a morphism $\mathsf{m}_{A,B} : ?(A ⅋ B) \to ?A ⅋ ?B$ which is a $⅋$-morphism. As before we will use the same notation for the dual morphism: $\mathsf{m}_{A,B} : !A \otimes !B \to !(A \otimes B)$.

[1] Following [20] one may weaken this condition and only suppose that the restriction of $U_\mathcal{C}$ to a subcategory of $⅋\text{-}\mathsf{Mon}(\mathcal{C})$ closed by $⅋$ and & has a left adjoint.

3.2. Building a control category

We will now construct from $\mathcal{C}$ a *control category* $\mathbb{K}_{\mathcal{C}}$ as defined by Selinger [25]. It is defined by:

- objects of $\mathbb{K}_{\mathcal{C}}$ are $\parr$-monoids;
- given two $\parr$-monoids M and N, we set $\mathbb{K}_{\mathcal{C}}(M, N) = \mathcal{C}(!\bar{M}, \bar{N})$.

Informally one may think of $\mathbb{K}_{\mathcal{C}}$ as the co-Kleisli of the subcategory of $\mathcal{C}$ consisting in $\parr$-monoids. Just as for the co-Kleisli of $\mathcal{C}$ one obtains that $\mathbb{K}_{\mathcal{C}}$ is a CCC: composition is defined using the ! comonad and digging as usual, the cartesian product is the & and the closeness is obtained by noting that $N^M = ?_{\mathcal{C}}(\bar{M}^\perp) \parr N$ is a $\parr$-monoid.

The natural transformation ϕ. For each pair of $\mathcal{C}$-objects A, B we define the $\mathcal{C}$-morphism $\phi_{A,B} : !(A \parr B) \to !A \parr ?B$ to be the transposed of

$$!(A \parr B) \otimes !B^\perp \xrightarrow{\mathsf{m}} !((A \parr B) \otimes B^\perp) \xrightarrow{!\varepsilon} !A$$

Equivalently we have:

$$\phi_{A,B} = !(A \parr B) \xrightarrow{\tilde{\mathsf{m}}} !((A \parr B) \otimes B^\perp) \parr ?B \xrightarrow{!\varepsilon \parr ?B} !A \parr ?B$$

By construction we get that $\phi_{A,B}$ is natural in A. As a consequence of properties of transposition and of m, it is also natural in B.

Interpreting promotion. The natural transformation ϕ should be understood as a classical version of the transformation m. It is used to interpret the promotion rule as follows: if $f : G \to A \parr D$ then we define $f^! : !G \to !A \parr ?D$ by:

$$f^! = !G \xrightarrow{!f} !(A \parr D) \xrightarrow{\phi_{A,D}} !A \parr ?D.$$

Polarized promotion. Given a $\mathcal{C}$-object A and a $\parr$-monoid N we define the $\mathcal{C}$-morphism $\psi_{A,N} : !(A \parr \bar{N}) \to !A \parr \bar{N}$ as:

$$\psi_{A,N} = !(A \parr \bar{N}) \xrightarrow{\phi_{A,\bar{N}}} !A \parr ?\bar{N} \xrightarrow{!A \parr \mathsf{alg}_N} !A \parr \bar{N}$$

It is worth noting that $\psi_{A,N}$ is the transposed of:

$$!(A \parr \bar{N}) \otimes \bar{N}^\perp \xrightarrow{\mathsf{Id} \otimes \mathsf{coalg}_{N^\perp}} !(A \parr \bar{N}) \otimes !\bar{N}^\perp \xrightarrow{\mathsf{m}} !((A \parr \bar{N}) \otimes !\bar{N}^\perp) \xrightarrow{!\varepsilon} !A$$

By construction ψ is natural in A.

Just as ϕ is used to interpret functorial promotion, ψ is used to interpret *polarized* promotion: if $f : \bar{P} \to A \parr \bar{N}$ where P is a $\otimes$-comonoid and N is a $\parr$-monoid, we define $f^! : \bar{P} \to !A \parr \bar{N}$ by

$$\bar{P} \xrightarrow{\mathsf{coalg}} !\bar{P} \xrightarrow{!f} !(A \parr \bar{N}) \xrightarrow{\psi} !A \parr \bar{N}$$

The $\parr_K$ functor. Let N be a fixed $\parr$-monoid. We are now able to define the functor $\_ \parr_K N$ on $\mathbb{K}_{\mathcal{C}}$ by:

- for each $\parr$-monoid A, $A \parr_K N = A \parr N$;
- if $f : !\bar{A} \to_{\mathcal{C}} \bar{B}$ (that is $f : A \to_{\mathbb{K}_{\mathcal{C}}} B$) we set
$$\begin{aligned} f \parr_K N &= !(\bar{A} \parr \bar{N}) \xrightarrow{\phi_{\bar{A},\bar{N}}} !\bar{A} \parr ?\bar{N} \xrightarrow{f \parr \mathsf{alg}_N} \bar{B} \parr \bar{N} \\ &= !(\bar{A} \parr \bar{N}) \xrightarrow{\psi_{\bar{A},N}} !\bar{A} \parr \bar{N} \xrightarrow{f \parr \bar{N}} \bar{B} \parr \bar{N} \end{aligned}$$
so that $f \parr_K N : A \parr N \to_{\mathbb{K}_{\mathcal{C}}} B \parr N$ as expected.

Using the naturality of ψ, the functoriality of $\_ \parr_K N$ is consequence of the commutation of the following two diagrams:

$$\begin{tikzcd} !(\bar{A} \parr \bar{N}) \arrow[r, "\mathsf{der}"] \arrow[d, "\psi"'] & \bar{A} \parr \bar{N} \\ !\bar{A} \parr \bar{N} \arrow[ur, "\mathsf{der} \parr \bar{N}"'] & \end{tikzcd} \qquad \begin{tikzcd} !(\bar{A} \parr \bar{N}) \arrow[r, "\mathsf{dig}"] \arrow[dd, "\psi"'] & !!(\bar{A} \parr \bar{N}) \arrow[d, "!\psi"] \\ & !(!\bar{A} \parr \bar{N}) \arrow[d, "\psi"] \\ !\bar{A} \parr \bar{N} \arrow[r, "\mathsf{dig} \parr \bar{N}"] & !!\bar{A} \parr \bar{N} \end{tikzcd}$$

These diagrams may be proved by transposing them and using the properties of $\mathcal{C}$. It should be noted that they express invariance of the categorical semantics w.r.t. the dereliction and commutative cut eliminations in polarized LL.

Control categories. We roughly recall the definition of control categories, see [25] for the detailed one: a control category is a distributive, symmetric premonoidal category with codiagonals which is also cartesian closed and such that the canonical morphism $s_{A,B,C} : B^A \parr C \to (B \parr C)^A$ is a natural isomorphism in A, B and C satisfying some coherence conditions.

We already have seen that $\mathbb{K}_{\mathcal{C}}$ is a CCC. The symmetric premonoidal structure is given by the definition of $\parr_K$. As for the co-Kleisli of $\mathcal{C}$, we have a functor which embeds $\parr\text{-}\mathsf{Mon}(\mathcal{C})$ into $\mathbb{K}_{\mathcal{C}}$, which associates to any $\parr$-morphism $f : M \to N$ the central morphism:

$$!\bar{M} \xrightarrow{\mathsf{der}} \bar{M} \xrightarrow{f} \bar{N}$$

In particular codiagonals in $\mathbb{K}_{\mathcal{C}}$ are given by the images of w and c by this embedding. Also the distributivity of $\mathbb{K}_{\mathcal{C}}$ (essentially the fact that $\_ \parr_K N$ preserves finite products) is consequence of the distributivity $\parr\text{-}\mathsf{Mon}(\mathcal{C})$-isomorphism: $A \parr (B \mathbin{\&} C) \doteq (A \parr B) \mathbin{\&} (A \parr C)$.

Selinger calls *exponential strength* the morphism $s_{A,B,C}$. In $\mathbb{K}_{\mathcal{C}}$, B^A is $?_{\mathcal{C}}\bar{A}^\perp \parr B$ so that $s_{A,B,C}$ is just built from the associativity isomorphism for $\parr$. More precisely:

$$s_{A,B,C} = !((?\bar{A}^\perp \parr \bar{B}) \parr \bar{C}) \xrightarrow{\mathsf{der}} (?\bar{A}^\perp \parr \bar{B}) \parr \bar{C} \xrightarrow{\sim} ?\bar{A}^\perp \parr (\bar{B} \parr \bar{C})$$

The coherence conditions are immediately checked.

Theorem 3.1 *Let $\mathcal{C}$ be a Lafont category. Then $\mathbb{K}_{\mathcal{C}}$ is a control category. In particular $\mathbb{K}_{\mathcal{C}}$ is a model of* LLP *as defined in [16].*

In particular when applying this construction to Girard's coherent spaces, one gets back the so-called *correlation spaces* that were used to define the semantics of LC [9]. Actually the whole construction above is the categorical abstraction of this particular concrete case.

4. CPS-Translations

As explained in the introduction we use Parigot's lambda-mu calculus for encoding classical logic. We use lambda-calculus as the target language of the CPS-translations. However, in the spirit of [2], we will design a typing system (à la Curry) which allows to express some linearity constraints; more precisely we add to ordinary intuitionistic types a linear and a non linear negation that are used to type the (translations of) continuations. This typing system may be viewed as a fragment of the linear lambda-calculus DILL [1] where the negation $\neg A$ is a shorthand for $A \to \bot$ whereas $\neg_0 A$ is $A \multimap \bot$.

We use the standard notations for contexts of typed variables; in particular Γ, Γ' denotes the union of two contexts and $\Gamma \setminus x : A$ is the context Γ from which the declaration $x : A$ has been removed if it was there.

4.1. The lambda-mu-calculus

Let be given two denumerable disjoint sets respectively of λ-variables (denoted x, y, etc.) and μ-variables (denoted α, β, etc.). Lambda-mu-terms are constructed by the usual lambda-calculus operations (λ-variable, application denoted $(u)v$ and abstraction) together with:

naming: $[\alpha]t$;

mu abstraction: $\mu\alpha\, t$;

where the μ-variable α is bound in the abstraction case. A term of the form $[\alpha]t$ is called a *named term*.

The two reduction rules for CBN lambda-mu-calculus are:

$$(\lambda x\, t)u \to_\beta t[u/x]$$
$$(\mu\alpha\, t)u \to_\mu \mu\alpha\, t[[\alpha](v)u/[\alpha]v]$$

We use the notation $t \to t'$ if t can be reduced in t' by one step of β- or μ-reduction.

Types are the atomic types (X, Y, ...) and the arrow types ($A \to B$). Typing judgments are sequents of the form $\Gamma \vdash u : A, \Delta$ or $\Gamma \vdash u, \Delta$ where Γ (resp. Δ) contains type declarations for (at least) all the λ-variables free in u (resp. all the μ-variables). The second case of judgment, the one where u has no type, may be understood as a shorthand for $u : \bot$; we write $u : \Theta$ where Θ may be an empty type when we don't want to specify whether u is typed or not in a judgment. The derivation rules are:

$$\frac{}{x : A \vdash x : A}$$

$$\frac{\Gamma \vdash u : B, \Delta}{\Gamma \setminus \{x : A\} \vdash \lambda x\, u : A \to B, \Delta}$$

$$\frac{\Gamma \vdash t : A \to B, \Delta \qquad \Gamma' \vdash u : A, \Delta'}{\Gamma, \Gamma' \vdash (t)u : B, \Delta, \Delta'}$$

$$\frac{\Gamma \vdash t : A, \Delta}{\Gamma \vdash [\alpha]t, \alpha : A, \Delta \setminus \{\alpha : A\}}$$

$$\frac{\Gamma \vdash t, \Delta}{\Gamma \vdash \mu\alpha\, t : A, \Delta \setminus \{\alpha : A\}}$$

4.2. The continuation lambda-calculus CLC

Terms of CLC are the usual lambda-terms in which we indifferently use lambda and mu-variables. Differences arise with types which are augmented with: negation types ($\neg A$) and linear negation types ($\neg_0 A$).

The typing judgments are sequents of the form $\Xi \vdash t : A$ or $\Xi \vdash t$ where Ξ is a context of the form $x_1 : A_1, \ldots, x_n : A_n$ or $x_1 : A_1, \ldots, x_n : A_n \mid z : A_0$. The second case of context will be said to contain a *linear part* $z : A_0$. A context in which it is not specified whether it contains or not a linear part will be denoted $\Gamma \mid \Pi$.

We will further restrict types by asking that in a judgment only the top-level formulas (that is A and the A_i's) may have the shape $\neg_0 B$, all the inner occurrences of $\neg_0$ in the sequent must be immediately lying under a $\neg$. This constraint which is satisfied by the (to be defined) Plotkin's and Krivine's translations will be useful to faithfully embed CLC into LL_{pol}.

The typing rules are given in figure 1 where in all the rules the variable possibly occurring in Π is distinct from any variable in Γ and Δ and in the $\neg_0$-elim, at least one of Π and Π' is empty.

4.3. Plotkin's CBN translation

This translation is definable on untyped lambda-mu terms as follows:

$$x^\bullet = \lambda k\, (x)k$$
$$(\lambda x\, u)^\bullet = \lambda k\, (k)\lambda x\, u^\bullet$$
$$((u)v)^\bullet = \lambda k\, (u^\bullet)\lambda m\, ((m)v^\bullet)k$$
$$(\mu\alpha\, u)^\bullet = \lambda\alpha\, u^\bullet$$
$$([\alpha]u)^\bullet = (u^\bullet)\alpha$$

$$
\frac{}{\mid x : A \vdash x : A} \qquad \frac{\Gamma \mid x : A \vdash t : \Theta}{\Gamma, x : A \vdash t : \Theta}
$$

$$
\frac{\Gamma \mid z : A \vdash t}{\Gamma \vdash \lambda z\, t : \neg_0 A} \qquad \frac{\Gamma \mid \Pi \vdash t : \neg_0 A \qquad \Delta \mid \Pi' \vdash u : A}{\Gamma, \Delta \mid \Pi \cup \Pi' \vdash (t)u}
$$

$$
\frac{\Gamma \mid \Pi \vdash t}{\Gamma \setminus \{x : A\} \mid \Pi \vdash \lambda x\, t : \neg A} \qquad \frac{\Gamma \mid \Pi \vdash t : \neg A \qquad \Delta \vdash u : A}{\Gamma, \Delta \mid \Pi \vdash (t)u}
$$

$$
\frac{\Gamma \mid \Pi \vdash t : B}{\Gamma \setminus \{x : A\} \mid \Pi \vdash \lambda x\, t : A \to B} \qquad \frac{\Gamma \mid \Pi \vdash t : A \to B \qquad \Delta \vdash u : A}{\Gamma, \Delta \mid \Pi \vdash (t)u : B}
$$

Figure 1. Typing CLC

The idea is to encode continuations as lists of arguments (as we are in CBN) using some list encoding in lambda-calculus; in particular the term $(x)u_1 \ldots u_n$ is translated as:

$$\lambda k\,(x)\lambda k_1\,((k_1)u_1)\lambda k_2\,((k_2)u_2)\ldots\lambda k_n\,((k_n)u_n)k$$

up to some easy reductions ("easy" here may be understood as "linear").

The translation may be extended to types; firstly we define the type translation:

$$
\begin{aligned}
X^\bullet &= X \\
(A \to B)^\bullet &= \neg\neg_0 A^\bullet \to \neg\neg_0 B^\bullet
\end{aligned}
$$

and finally typing judgments are translated as:

$$(\Gamma \vdash t : \Theta, \Delta)^\bullet = \neg\neg_0\Gamma^\bullet, \neg_0\Delta^\bullet \vdash t^\bullet : \neg\neg_0\Theta^\bullet$$

with the convention that if Θ is the empty type then so is $\neg\neg_0\Theta^\bullet$. Note that, as we announced, our restriction on $\neg_0$-types is satisfied by this translation.

It is straightforward to check that the type translation rules are compatible with the term ones.

4.4. Krivine's translation

Krivine's translation [12, 13] is essentially a slight amelioration of the former Gödel's translation of classical into intuitionistic logic; contrarily to Plotkin's one it is type dependent. A CLC type A^* is associated to a lambda-mu type A by the following rules:

$$
\begin{aligned}
X^* &= \neg X \\
(A \to B)^* &= A^* \to B^*
\end{aligned}
$$

The two kinds of typing judgments in lambda-mu are translated as follows:

$$
\begin{aligned}
(\Gamma \vdash t : A, \Delta)^* &= \Gamma^*, \neg_0\Delta^*, k : \neg_0 A^* \vdash t^* \\
(\Gamma \vdash t, \Delta)^* &= \Gamma^*, \neg_0\Delta^* \vdash t^*
\end{aligned}
$$

where in the first form k is a variable not occurring in Γ, Δ. It is immediate to check that the $\neg_0$-type restriction is satisfied. The translation is defined by induction on the type derivation of t as follows:

$$
\begin{aligned}
(x^A : A)^* &= (k^{\neg_0 A^*})x^{A^*} \\
(\lambda x^A\, t^B : A \to B)^* &= (k^{\neg_0(A^* \to B^*)})\lambda x^{A^*}\,(C_B)\lambda k^{\neg_0 B^*}\, t^* \\
((u^{A \to B})v^A : B)^* &= \\
&\quad (k^{\neg_0 B^*})((C_{A \to B})\lambda k^{\neg_0(A^* \to B^*)}\, u^*)(C_A)\lambda k^{\neg_0 A^*}\, v^* \\
(\mu\alpha^A\, u : A)^* &= u^*[k^{\neg_0 A^*}/\alpha^{\neg_0 A^*}] \\
([\alpha^A]u^A)^* &= u^*[\alpha^{\neg_0 A^*}/k^{\neg_0 A^*}]
\end{aligned}
$$

where the C_A are terms of type $\neg\neg_0 A^* \to A^*$ defined by induction on A as follows:

$$
\begin{gathered}
C_X = \lambda n_0^{\neg\neg_0\neg X}\,\lambda x^X\,(n_0)\lambda n_1^{\neg X}\,(n_1)x \\
C_{A \to B} = \\
\lambda n^{\neg\neg_0(A^* \to B^*)}\,\lambda a^{A^*}\,(C_B)\lambda k^{\neg_0 B^*}\,(n)\lambda f^{A^* \to B^*}\,(k)(f)a
\end{gathered}
$$

One easily shows by induction on n that if $A = A_1 \to \cdots \to A_n \to X$ (so that $A^* = A_1^* \to \cdots \to A_n^* \to \neg X$) then

$$C_A = \lambda n^{\neg\neg_0 A^*}\,\lambda a_1^{A_1^*}\ldots\lambda a_n^{A_n^*}\,\lambda x^X\,(n)\lambda f^{A^*}\,(f)a_1\ldots a_n x$$

4.5. Translations of CLC and lambda-mu into polarized LL

We first define the translation of types. Lambda-mu types are translated as LLP formulas and CLC types as $\mathsf{LL}_{\mathrm{pol}}$ formulas. We take advantage that lambda-mu types form a subset of CLC types and that LLP formulas form a subset of $\mathsf{LL}_{\mathrm{pol}}$ formulas to factorize these two translations into the following:

$$
\begin{array}{rcccl}
X^- & = & X & & \\
(A \to B)^- & = & !A^- \multimap B^- & = & ?(A^-)^\perp \invamp B^- \\
(\neg A)^- & = & ?(A^-)^\perp & & \\
(\neg_0 A)^- & = & (A^-)^\perp & &
\end{array}
$$

Note that the type $\neg_0 A$ is translated as a positive formula; due to our restriction on CLC types all the other ones are translated as negative formulas. Also the reason for allowing weakly polarized formulas in $\mathsf{LL}_{\mathrm{pol}}$ is that the CLC type $\neg\neg_0 A$ is translated as $?A^-$ where by definition A^- is already negative.

Similarly we define in one shot the judgment translations: typing judgments are translated as linear sequents: $(\Gamma \mid \Pi \vdash u : \Theta, \Delta)^- = \vdash ?(\Gamma^-)^\perp, (\Pi^-)^\perp, \Theta^-, \Delta^-$ where Δ is empty in the CLC case and Π is empty in the lambda-mu case.

CLC. The translations of type derivations into sequent calculus proofs in $\mathsf{LL}_{\mathrm{pol}}$ are given in figure 2.

Lambda-mu-calculus. The lambda-calculus rules (variable, abstraction, application) are translated as in the CLC case (with Π always empty and adding a context Δ on each right member of sequents) and the mu-rules are given by:

$$([\alpha^A]t^A)^- = \dfrac{\begin{array}{c}\vdots\\ \vdash ?(\Gamma^-)^\perp, A^-, \Delta^-\end{array}}{\vdash ?(\Gamma^-)^\perp, A^-, \Delta^- \setminus \{A^-\}}$$

$$(\mu\alpha^A\, t : A)^- = \dfrac{\begin{array}{c}\vdots\\ \vdash ?(\Gamma^-)^\perp, \Delta^-\end{array}}{\vdash ?(\Gamma^-)^\perp, A^-, \Delta^- \setminus \{A^-\}}$$

4.6. Linear analysis of CPS-translations

The embedding of lambda-systems into polarized linear systems allows to also embed the CPS-translations. Plotkin's translation is shown to correspond to the box translation $(.)^b$ of LLP into $\mathsf{LL}_{\mathrm{pol}}$, and Krivine's translation to the reversing one $(.)^\rho$. We therefore get:

Theorem 4.1 *Up to some axiom reductions and some η-expansions the following diagrams commute:*

$$\begin{array}{ccc} \lambda\mu & \xrightarrow{(.)^-} & \mathsf{LLP} \\ {\scriptstyle(.)^\bullet}\downarrow & & \downarrow{\scriptstyle(.)^b} \\ \mathsf{CLC} & \xrightarrow{(.)^-} & \mathsf{LL}_{\mathrm{pol}} \end{array} \qquad \begin{array}{ccc} \lambda\mu & \xrightarrow{(.)^-} & \mathsf{LLP} \\ {\scriptstyle(.)^*}\downarrow & & \downarrow{\scriptstyle(.)^\rho} \\ \mathsf{CLC} & \xrightarrow{(.)^-} & \mathsf{LL}_{\mathrm{pol}} \end{array}$$

The proof relies on the following facts:

- For any lambda-mu type A we have $A^{-^b} = ?A^{\bullet -}$.
- For any lambda-mu judgment $\Gamma \vdash t : \Theta, \Delta$ we have $(\Gamma \vdash t : \Theta, \Delta)^{-^b} = (\Gamma \vdash t : \Theta, \Delta)^{\bullet -}$.
- For any lambda-mu type A we have $A^{-^\rho} = A^{*-}$.
- Recall that ρ_N is the "canonical" proof of $\vdash\, !N^\perp, N$, that is the syntactic version of the ?-algebra structure of N. For any lambda-mu type A we have $(C_A)^- = \rho_{A^{*-}}$, up to some axiom reductions.

5. Syntax and semantics

Selinger [25] has given a canonical construction of control categories as $R^{\mathcal{C}}$ where $(\mathcal{C}, R)$ is a suitable *continuation category* (see [11]). We denote by $[\![t]\!]_{R^{\mathcal{C}}}$ the call by name interpretation of the lambda-mu-term t in the control category $R^{\mathcal{C}}$ (as defined by Selinger) and by $[\![t]\!]_{(\mathcal{C},R)}$ the interpretation of the target term t in $(\mathcal{C}, R)$. With these notations, and using the fact that Hofmann-Streicher's translation coincides, in our setting, with Plotkin's one, we may reformulate one of Selinger's results (deduced from Hofmann-Streicher's work) as:

Theorem 5.1 (Hoffman-Streicher / Selinger) *Let t be a lambda-mu-term, then $[\![t]\!]_{R^{\mathcal{C}}} = [\![t^\bullet]\!]_{(\mathcal{C},R)}$.*

This gives the relation between Plotkin's syntactical translation and the $R^{\mathcal{C}}$ construction of a control category. We can do the same analysis for Krivine's translation. If $[\![t]\!]_{\mathbb{K}_{\mathcal{C}}}$ is Selinger's call by name interpretation of the lambda-mu-term t in $\mathbb{K}_{\mathcal{C}}$ and $[\![t]\!]_{\mathcal{C}}$ is the natural interpretation of the CLC-term t in the Lafont category $\mathcal{C}$, we get:

Lemma 5.1 *If $u : \neg\neg_0 A^*$ is a CLC-typed lambda-term, then $[\![(C_A)u]\!]_{\mathcal{C}} = [\![u]\!]_{\mathcal{C}}; \mathsf{alg}_{A^*}$*

Theorem 5.2 *Let t be a lambda-mu-term, $[\![t]\!]_{\mathbb{K}_{\mathcal{C}}} = [\![(C)\lambda k\, t^*]\!]_{\mathcal{C}}$.*

Proof: By induction, we consider each typing rule for the lambda-mu-calculus:

- for a contraction or weakening rule, we use the monoidality of alg_A and lemma 5.1;
- for the logical rules, we apply the induction hypothesis, lemma 5.1 and $\mathsf{der}_A; \mathsf{alg}_A = \mathsf{Id} A$. $\square$

6. Conclusion

We have shown how to establish a correspondence between call by name CPS-translations and the translations from LLP to $\mathsf{LL}_{\mathrm{pol}}$. This can be extended to the call by value case, by a positive embedding (type $\mapsto$ positive formula) of the lambda-mu-calculus in LLP. In particular for Plotkin's call by value translation [23], we obtain the following diagram:

$$\begin{array}{ccc} \lambda\mu & \xrightarrow{(.)^+} & \mathsf{LLP} \\ {\scriptstyle(.)^\bullet}\downarrow & & \downarrow{\scriptstyle(.)^b} \\ \mathsf{CLC} & \xrightarrow{(.)^+} & \mathsf{LL}_{\mathrm{pol}} \end{array}$$

$$(x^A : A)^- = \dfrac{}{\vdash (A^-)^\perp, A^-}$$

$$\left(\dfrac{\Gamma \mid x : A \vdash t : \Theta}{\Gamma, x : A \vdash t : \Theta}\right)^- = \dfrac{\begin{array}{c}\vdots \\ \vdash ?(\Gamma^-)^\perp, (A^-)^\perp, \Theta^-\end{array}}{\vdash ?(\Gamma^-)^\perp, ?(A^-)^\perp, \Theta^-}$$

$$(\lambda x^A\, t^B : A \to B)^- = \dfrac{\begin{array}{c}\vdots \\ \vdash ?(\Gamma^-)^\perp, (\Pi^-)^\perp, B^-\end{array}}{\vdash ?(\Gamma^-)^\perp \setminus \{?(A^-)^\perp\}, (\Pi^-)^\perp, ?(A^-)^\perp \parr B^-}$$

$$(\lambda x^A\, t : \neg A)^- = \dfrac{\begin{array}{c}\vdots \\ \vdash ?(\Gamma^-)^\perp, (\Pi^-)^\perp\end{array}}{\vdash ?(\Gamma^-)^\perp \setminus \{?(A^-)^\perp\}, (\Pi^-)^\perp, ?(A^-)^\perp}$$

$$(\lambda x^A\, t : \neg_0 A)^- = \begin{array}{c}\vdots \\ \vdash ?(\Gamma^-)^\perp, (A^-)^\perp\end{array}$$

$$((t^{A \to B}) u^A : B)^- = \dfrac{\begin{array}{c}\vdots \\ \vdash ?(\Gamma^-)^\perp, (\Pi^-)^\perp, ?(A^-)^\perp \parr B^-\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}\vdots \\ \vdash ?(\Gamma'^-)^\perp, A^-\end{array}}{\vdash ?(\Gamma'^-)^\perp, !A^-} \qquad \dfrac{}{\vdash (B^-)^\perp, B^-}}{\vdash ?(\Gamma'^-)^\perp, !A^- \otimes (B^-)^\perp, B^-}}{\vdash ?(\Gamma^-)^\perp, ?(\Gamma'^-)^\perp, (\Pi^-)^\perp, B^-}$$

$$((t^{\neg A}) u^A)^- = \dfrac{\begin{array}{c}\vdots \\ \vdash ?(\Gamma^-)^\perp, (\Pi^-)^\perp, ?(A^-)^\perp\end{array} \qquad \dfrac{\begin{array}{c}\vdots \\ \vdash ?(\Gamma'^-)^\perp, A^-\end{array}}{\vdash ?(\Gamma'^-)^\perp, !A^-}}{\vdash ?(\Gamma^-)^\perp, ?(\Gamma'^-)^\perp, (\Pi^-)^\perp}$$

$$((t^{\neg_0 A}) u^A)^- = \dfrac{\begin{array}{c}\vdots \\ \vdash ?(\Gamma^-)^\perp, (\Pi^-)^\perp, (A^-)^\perp\end{array} \qquad \begin{array}{c}\vdots \\ \vdash ?(\Gamma'^-)^\perp, (\Pi'^-)^\perp, A^-\end{array}}{\vdash ?(\Gamma^-)^\perp, ?(\Gamma'^-)^\perp, ((\Pi \cup \Pi')^-)^\perp}$$

Figure 2. Translation of CLC **into** $\mathsf{LL}_{\mathsf{pol}}$

A call by value analogue of Krivine's translation is possible to define but would require to replace CLC by a system with at most one formula on the left which is less natural.

This approach maps the CPS translations (from classical to intuitionistic logic) to the two translations $(.)^b$ and $(.)^\rho$ in linear logic. A natural question is to check whether this holds for any CPS translation. If it is the case, which is our current opinion, then the whole set of CPS translations can be classified into only two classes showing that many variations in the definitions of CPS translations are innocuous; e.g. there is no difference between Plotkin's translation and Hofmann-Streicher one when viewing them as translations of LLP into LL. It could also be that there are CPS translations the image of which are neither $(.)^b$ nor $(.)^\rho$, which would yield some new translation of LLP into LL_{pol}. Although we doubt such a new translation exists, it would certainly be very interesting for the theory of polarized linear systems if one eventually shows up.

An interesting remaining question is to study CPS translations in the framework of game semantics, in particular using the first author game model for LLP [17]. This could yield some similar relation between game semantics for classical logic and abstract machines as the ones studied in [5] for intuitionistic logic. There is also the work by Jim Laird [15] mentioned in the introduction, that introduces a special kind of game models for CPS translations. The question remains open whether this game model is related to the first author's one for LLP.

References

[1] A. Barber and G. Plotkin. Dual intuitionistic linear logic. Technical report, LFCS, University of Edinburgh, 1997.

[2] J. Berdine, P. O'Hearn, U. Reddy, and H. Thielecke. Linear continuation-passing. *Higher-Order and Symbolic Computation*, 15, 2002.

[3] G. Bierman. *On Intuitionistic Linear Logic*. PhD thesis, University of Cambridge, Computer Laboratory, 1993. Available as Technical Report 346, August 1994.

[4] G. Bierman. What is a categorical model of Linear Logic? In M. Dezzani and G. Plotkin, editors, *Proceedings of the Second International Conference on Typed Lambda Calculi and Applications*, volume 902 of *Lecture Notes in Computer Science*. Springer-Verlag, Apr. 1995.

[5] V. Danos, H. Herbelin, and L. Regnier. Games semantics and abstract machines. In *Proceedings of the 11^{th} Symposium on Logic in Computer Science*, New Brunswick, 1996. IEEE Computer Society Press.

[6] V. Danos, J.-B. Joinet, and H. Schellinx. LKQ and LKT: Sequent calculi for second order logic based upon dual linear decompositions of classical implication. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1995.

[7] V. Danos, J.-B. Joinet, and H. Schellinx. A new deconstructive logic: Linear logic. *Journal of Symbolic Logic*, 62(3):755–807, 1997.

[8] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[9] J.-Y. Girard. A new constructive logic: classical logic. *Mathematical Structures in Computer Science*, 1(3):225–296, 1991.

[10] T. Griffin. A formulae-as-types notion of control. In *Proceedings of the 1990 Principles of Programming Languages Conference*, pages 47–58. IEEE Computer Society Press, 1990.

[11] M. Hofmann and T. Streicher. Continuation models are universal for lambda-mu-calculus. In *Proceedings of the 12^{th} Symposium on Logic in Computer Science*, pages 387–397, Warsaw, 1997. IEEE Computer Society Press.

[12] J.-L. Krivine. *Lambda-Calcul : Types et Modèles*. Études et Recherches en Informatique. Masson, 1990.

[13] J.-L. Krivine. A general storage theorem for integers in call-by-name λ-calculus. *Theoretical Computer Science*, 129:79–94, 1994.

[14] Y. Lafont. The linear abstract machine. *Theoretical Computer Science*, 59:157–180, 1988.

[15] J. Laird. A game semantics of linearly used continuations. In A. Gordon, editor, *Foundations of Software Science and Computation Structures*, volume 2620 of *Lecture Notes in Computer Science*, pages 313–327. Springer-Verlag, 2003.

[16] O. Laurent. *Étude de la polarisation en logique*. Thèse de doctorat, Université Aix-Marseille 2, 2002.

[17] O. Laurent. Polarized games (extended abstract). In *Proceedings of the seventeenth annual IEEE symposium on Logic In Computer Science*, pages 265–274. IEEE Computer Society Press, July 2002.

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

[19] S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.

[20] P.-A. Melliès. Categorical models of linear logic revisited. Available at http://www.pps.jussieu.fr/∼mellies/papers/catmodels.ps.gz, 2002.

[21] C. Murthy. An evaluation semantics for classical proofs. In *Proceedings of the 6^{th} Symposium on Logic in Computer Science*, pages 96–107. IEEE Computer Society Press, 1991.

[22] M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Proceedings of Internationnal Conference on Logic Programming and Automated Deduction*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer-Verlag, 1992.

[23] G. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1:125–159, 1975.

[24] M. Quatrini and L. Tortora de Falco. Polarisation des preuves classiques et renversement. *Compte-Rendu à l'Académie des Sciences*, 323:113–116, 1996.

[25] P. Selinger. Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, 11(2):207–260, 2001.