From delimited CPS to polarisation

Guillaume Munch-Maccagnoni

From delimited CPS to polarisation

Guillaume Munch-Maccagnoni*

Abstract *The understanding of continuation-passing style (CPS) translations, an historical source of denotational semantics for programming languages, benefits from notions brought by linear logic [Gir87], such as focalisation, polarities and the involutive negation. Here we aim to show how linear logic helps as well when continuations are delimited, i.e. return and can be composed, in the sense of Danvy and Filinski [DF89].*

First we provide a polarised calculus with delimited control (first-class delimited continuations) which is, at the level of computation, a variant of Girard's polarised classical logic LC *[Gir91]. It contains variants of delimited control calculi which came as answers to the question "what order of evaluation can we consider with delimited control?". Thus our polarised calculus is one answer which is unifying to some degree.*

Subsequently we decompose the calculus through polarised linear logic. The only difference with non-delimited continuations is the use of specific exponentials, that account for the specific semantics of the target of delimited CPS translation, as well as for annotations of type-and-effect systems.

As a by-product, we obtain an explanation of CPS translations through a factoring, each step of which accounts for distinct phenomena of CPS translations. Although the factoring also holds for non-delimited CPS translations, it did not appear in its entirety before.

Contents

*Laboratoire PPS, Université Paris Diderot. April 20th, 2011.

Introduction

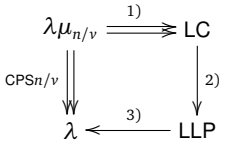
Continuation-passing style (CPS) translations are a paradox in the theory of programming languages, because they show that the λ calculus, this purely functional, idealised paradigm of computation, can be diverted so as to seemingly contradict the purely functional paradigm:

1. CPS translations give firm grounds for the notion of evaluation order, by "hard-coding" the order in the structure of the resulting term (Plotkin [Plo75]).
2. They give semantics of non-functional operators like call/cc.
3. Using continuation-passing it is possible to manipulate the output of the computation, not only the input. This highlights the possible symmetry between the two notions (see e.g. Curien and Herbelin [CH00]), and thus further departs from the functional point of view.

Understanding which abstract structures are involved in these translations is fruitful: the call-by-value CPS translation for instance inspired no less than the use of strong monads to describe the semantics of effects in call-by-value programming languages (Moggi [Mog89]).

Monads however are not sufficient to describe all the existing CPS semantics, as they do not account for the symmetry inherent to CPS: for instance, the call-by-name CPS translation of Lafont, Reus and Streicher [LRS93] is categorically dual to the call-by-value one (Selinger [Sel01]) — which means that if we see computation from an external point of view, the dynamics of expressions in the one coincide with the dynamics of evaluation contexts in the other. Therefore, while some CPS semantics account for monadic constructions, some others account for comonadic ones.

Polarisation



A single structure which accounts for both of these dual constructions yet exists in proof theory, under the form of Girard's polarised classical logic (LC) [Gir91]. LC was invented to provide a constructivisation of classical logic that would retain the symmetries of classical logic — at the cost of relaxing other properties such as the associativity of composition [LQdF05]. The most important symmetry kept is the involutivity of negation, by which $\neg\neg A$ is not only equivalent to A, but isomorphic to it.

Here we are concerned about understanding how notions such as polarities and the involutive negation of LC, once endowed with the proper computational interpretation, give a better view of what we can encode with CPS translations.

To do so, we show how CPS translations can be factored through LC, and isolate the three above aspects in successive steps.

Step 1) explains strictness (as in call-by-value) and laziness (or demand-driven-ness, as in call-by-name rather than call-by-need) of a λ calculus with control operators (such as Parigot's μ [Par92]) as the result of a choice, for the constructs of a language, of polarities of LC. Conversely this allows us to see LC through Curry-Howard as a programming language with (call/cc-related) effects that mixes demand-driven computations and data-driven computations (an interpretation that we introduced in [Mun09]).

Step 2) is the standard translation from LC to Laurent's so-called Polarised Linear Logic (LLP) [Lau02]. We show how at the level of the computational interpretation it relates the polarities of LC to the ones of LLP. While in LC expressions of both polarities may carry effects, in LLP they take a lower-level form: positives denote primitive elements of data (such as *values*), while negatives are defined by how they *react* to such data. These primitive ingredients yield demand-driven and data-driven computations in a construction that generalises in a symmetric fashion the monadic semantics of call-by-value — the latter only accounting for the positive part of LC.

Step 3) translates the intuitionistic-looking LLP into the lambda-calculus. It introduces specificities of CPS translations that seem orthogonal to their understanding: "administrative" redexes appear, and a gross simplification is introduced on the structure of the polarity-changing negation. This has the consequence of blurring, from the point of view of CPS, the distinction between terms and evaluation contexts that LLP and LC can describe.

But we are also concerned with generalising polarisation to broader forms of effects, and to this effect we give a similar decomposition for delimited CPS translations.

Delimited CPS translations

Danvy and Filinski introduced in [DF89] the *shift* (S) and *reset* ($[\cdot]$) control operators in call-by-value λ calculus. The *shift* operator reifies the current evaluation context under the form of a continuation, and passes it to its argument, like the *call/cc* control operator. However, unlike *call/cc*, it only captures the context up to the nearest *reset* operator – this is why we speak of *delimited* continuations. Beside standard call-by-value reductions, it has the following reductions:

$$[E(\mathsf{S}(\lambda f.t))] \to [(\lambda f.t)(\lambda x.E(x))]$$
$$[V] \to V$$

Amongst other proposals of calculi with composable first-class continuations, the one of Danvy and Filinski met practical and theoretical success.

To name a few, a result of Filinski [Fil94] showed the considerable expressiveness of these operators: any monadic effect that is expressible — that is, whose laws can be expressed as pure functional terms — is available in direct-style through the use of *shift* and *reset*. For instance, with the following:

$$read \stackrel{\text{def}}{=} \lambda().\mathsf{S}(\lambda k.\lambda s.([k()]s))$$
$$write \stackrel{\text{def}}{=} \lambda s.\mathsf{S}(\lambda k.\lambda\_.([k()]s))$$

$[t]V$ computes like t in which a memory cell has been initialised to the value V. (We adapted this example from [HG08].)

Delimited control operators have also been used to model linguistic phenomena (see Shan [Sha04] for an introduction). And in logic itself, these operators have been used (in a limited fashion) to internalise Markov's principle into intuitionistic logic (Herbelin [Her10]).

Danvy and Filinski describe a denotational semantics under the form of a CPS translation. In this translation, continuations may return and be composed. As a consequence, contrarily to usual CPS translations, the order of evaluation of the target matters, and Danvy and Filinski chose a target in call-by-value. This translation is well understood from the point of view of monads – and the particular type system Danvy and Filinski provide on top of their calculus itself can be described by "annotated" monads (Wadler [Wad94]).

This monadic analysis of the translation and the type system is paralleled by our polarised analysis; and, as far as the positive (call-by-value) part is concerned, our contribution resides in our analysis of the target of Danvy-Filinski's CPS translation.

We found in the literature an analysis of this target that relies on a second CPS translation in order to account for its call-by-value semantics; and in this last translation it is important that continuations are used linearly (as remarked by Kameyama & Hasegawa for their completeness result [KH03]). Moreover, it is the composition of the two translations, or "double" CPS translation, that is studied in practice.

Here we see the target of Danvy-Filinski's (single) CPS translation through Girard's linear logic and the so-called "*boring*" translation [Gir87] based on the $!(P^\perp \parr Q)$ coding of implication. This translation indeed yields a semantics of call-by-value (Maraist et al. [MOTW94]), and more precisely, corresponds to Moggi's monadic model [Mog89] where the strong monad is commutative (see Benton & Wadler [BW95]). Thus our axiomatisation of the target corresponds to Moggi's computational calculus together with the following equation that accounts for commutativity:

$$\mathsf{let}\, x\, \mathsf{be}\, t\, \mathsf{in}\, (\mathsf{let}\, y\, \mathsf{be}\, u\, \mathsf{in}\, v) =_\eta \mathsf{let}\, y\, \mathsf{be}\, u\, \mathsf{in}\, (\mathsf{let}\, x\, \mathsf{be}\, t\, \mathsf{in}\, v)\,. \qquad (1)$$

We call this calculus $\lambda_v^\times$, because it turns out to be essentially an extension with pairs of Plotkin's call-by-value λ calculus, which already had implicit in its definition that the order of evaluation (left-to-right or right-to-left) of function application did not matter [Plo75]. With a clear definition of the target, we can focus our analysis on single rather than double CPS translations.

The cornerstone of our approach is that the "*boring*" semantics of call-by-value allows us to link up with the polarised approach of LLP and LC. This approach is based on the decomposition of the type of continuations into the exponential modality ! of LLP and the polarity-changing negation that comes from linear logic: $\neg_R P = !P^\perp$ (we write $\neg_R$ the intuitionistic negation).

Delimited CPS translations replace negation by implication, which can be understood in the context of CPS translations as an annotated negation modality: $\neg_Q P \stackrel{\text{def}}{=} P \to Q$. Now, this annotated negation modality can be seen through the "*boring*" translation as an annotated exponential:

$$\neg_Q P = !_Q P^\perp \stackrel{\text{def}}{=} !(P^\perp \parr Q)$$

(It is an exponential in the sense that $!_P$ is a functor that at least enjoys $!_P(A\&B) \simeq !_P A \otimes !_P B$, since $\parr$ distributes over &. The idea that we can consider various kinds of exponentials in linear logic is common, see e.g. Danos, Joinet and Schellinx [DJS93].)

Thus, we can imagine a variant of LLP (which we call $\mathsf{LLP}^{!_P ?_P}$) which is obtained from linear logic by replacing the usual exponential by the annotated modalities $?_P$ and $!_P$. Then on top of it we can build a variant of LC, whose computational interpretation is a calculus in which we can retrieve existing delimited control calculi.

We can see the fact that LLP lends itself to analogies with call-by-value — rather than call-by-name — λ calculus as a theoretical argument in favour of Danvy and Filinski's choice. Indeed, if the target of the CPS translation had been in call-by-name (as in Sabry's variant of the calculus [Sab96]), then we would have been unable to provide a polarised decomposition.

The interest of polarisation is that it allows us to account for CPS translations that are not directly related to the monadic approach, such as the semantics of call-by-name $\lambda\mu$ calculus and indeed of LC itself. We show that this is also the case with delimited control by decomposing a call-by-name variant of Danvy-Filinski's calculus that has been introduced in Herbelin & Ghilezan [HG08].

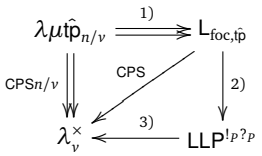
Method

The key technical aspect of this work is the use of a syntax that gives an independent status to evaluation contexts, with their own variables and binders, in the lineage of Curien and Herbelin's $\bar{\lambda}\mu\tilde{\mu}$ calculus [CH00], which can also be seen as a term assignment for sequent calculus. We introduced $\mathsf{L}_{\mathrm{foc}}$ in [Mun09] in order to refine the $\bar{\lambda}\mu\tilde{\mu}$ calculus to provide a computational interpretation of LC. We see such a notation as a tool for the mind, with which we strive to express the more properties of semantics with the fewest possible symbols.

Another key technical aspect is that we rely on a decomposition of *shift* and *reset* into operators that are simpler from the point of view of CPS translations: the μ operator of Parigot [Par92] together with a dynamically-scoped continuation variable written $\hat{\mathrm{tp}}$. This decomposition has been introduced with the $\lambda\mu\hat{\mathrm{tp}}_v$ calculus of Ariola, Herbelin and Sabry [AHS04] and Herbelin provided a syntax for it *à la* $\bar{\lambda}\mu\tilde{\mu}$ in [Her05].

Technical contribution

In Section 1, we produce a polarised calculus for delimited control that we call $\mathsf{L}_{\mathrm{foc},\hat{\mathrm{tp}}}$. It is a *symmetric* (yet confluent) variant of $\lambda\mu\hat{\mathrm{tp}}$ that extends $\mathsf{L}_{\mathrm{foc}}$. We describe its sound CPS semantics into $\lambda_v^{\times}$.



We show that this calculus contains both $\lambda\mu\hat{\mathrm{tp}}_v$ and its call-by-name variant $\lambda\mu\hat{\mathrm{tp}}_n$ [HG08], in the sense that the constructs of these languages can be defined in $\mathsf{L}_{\mathrm{foc},\hat{\mathrm{tp}}}$ and that these definitions induce translations – 1) – through which the CPS translations factor.

We also study the typed versions of these calculi and introduce a type-and-effect system for $\mathsf{L}_{\mathrm{foc},\hat{\mathrm{tp}}}$, extension of LC, with annotations of effects related to shifts of polarities. This type system decomposes the type-and-effect systems of $\lambda\mu\hat{\mathrm{tp}}_v$ and $\lambda\mu\hat{\mathrm{tp}}_n$.

In Section 2, we define a system that is essentially Laurent's LLP [Lau02] in which the exponentials enjoy a richer structure. We call the system $\mathsf{LLP}^{!_p?_p}$ (we omit additives, of which we have no use for our goal). We show how the CPS translation of $\mathsf{L}_{\mathrm{foc},\hat{\mathrm{tp}}}$ factors through $\mathsf{LLP}^{!_p?_p}$ in a manner that recalls the translation of LC into LLP: 2) & 3).

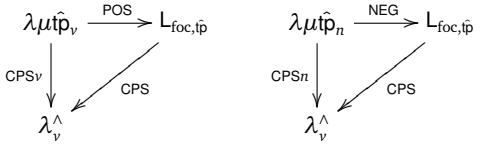
We explain the modalities $!_p$ and $?_p$ by describing a translation of $\mathsf{LLP}^{!_p?_p}$ into LL similar to the one described for LLP by Laurent [Lau02].

In Section 3 we compare our approach with extant works.

1 A polarised calculus for delimited control

We introduce in Fig. 1 a polarised calculus of delimited control ($\mathsf{L}_{\mathrm{foc},\hat{\mathrm{tp}}}$). The $\mathsf{L}_{\mathrm{foc}}$ part accounts for polarised classical logic (as in [Mun09]), while the $\hat{\mathrm{tp}}$ part account for the delimited control operators of the $\lambda\mu\hat{\mathrm{tp}}_v$ calculus [AHS04].

We describe its CPS semantics and we show that CPS translations of the (call-by-value) $\lambda\mu\hat{\mathrm{tp}}_v$ calculus of Ariola, Herbelin & Sabry and the (call-by-name) $\lambda\mu\hat{\mathrm{tp}}_n$ calculus of Herbelin & Ghilezan factor in the following manner:



where the top arrow relies moreover on the derivability of implication in $\mathsf{L}_{\mathrm{foc},\hat{\mathrm{tp}}}$ (modulo administrative reductions for the second diagram).

Proposition 1 (Confluence)**.** *The reduction* $\rightarrow$ *is confluent.*

Proof. The reduction $\rightarrow_0$ is defined by an orthogonal pattern rewrite system, which implies confluence for $\rightarrow$ (Nipkow [Nip91]). □

1.1 Understanding $\mathsf{L}_{\mathbf{foc},\hat{\mathrm{tp}}}$

1.1.1 Generalities on system L

Our syntax models computation as the rewriting of pairs $\langle t \,\|\, e\rangle$, also written c for *commands*. This rewriting is seen as the interaction between expressions (programs) $\langle t \mid$ and their context of evaluation $\mid e\rangle$. Thus they are kind of abstract machines states, but abstract ones in the sense that the syntax features variables and binders.

The first example of such a machine is Krivine's abstract machine. It has abstraction, application and a control operator as expressions, and stacks as evaluation contexts:

$$\langle t\, u \,\|\, E\rangle \rightarrow \langle t \,\|\, u \cdot E\rangle$$
$$\langle \lambda x.t \,\|\, u \cdot E\rangle \rightarrow \langle t\,[u/x] \,\|\, E\rangle$$
$$\langle \mu\alpha.c \,\|\, E\rangle \rightarrow c\,[E/\alpha]$$

This machine describes a call-by-name reduction strategy.

Curien and Herbelin [CH00] gave an equally elegant machine for computing in call-by-value: the $\bar{\lambda}\mu\tilde{\mu}_Q$ calculus. It only requires to introduce a new kind of binders on the right.

$$\langle t\, u \,\|\, E\rangle \rightarrow \langle t \,\|\, u \cdot E\rangle$$
$$\langle \lambda x.t \,\|\, u \cdot E\rangle \rightarrow \langle u \,\|\, \mu x.\langle t \,\|\, E\rangle\rangle$$
$$\langle V \,\|\, \mu x.c\rangle \rightarrow c\,[V/x]$$
$$\langle \mu\alpha.c \,\|\, E\rangle \rightarrow c\,[E/\alpha]$$

The $\langle \mu x.c \rangle$ binder must be understood as the "let x be $\Box$ in ..." context, as the one of Moggi's computational λ calculus [Mog89]. The second line reads:

$$\bar{E}[(\lambda x.t)\,u] \rightarrow \bar{E}[\mathsf{let}\, x\, \mathsf{be}\, [u]\, \mathsf{in}\, t]$$

if $\bar{E}$ is some term with a hole that corresponds to the evaluation context E.

One advantage of working with solid contexts is that the equational theory becomes simplified. For instance, the following equation of Moggi's calculus for x fresh in v:

$$\mathsf{let}\, x\, \mathsf{be}\, t\, \mathsf{in}\, (\mathsf{let}\, y\, \mathsf{be}\, u\, \mathsf{in}\, v) \simeq \mathsf{let}\, y\, \mathsf{be}\, (\mathsf{let}\, x\, \mathsf{be}\, t\, \mathsf{in}\, u)\, \mathsf{in}\, v$$

becomes superfluous since it can be derived from other rules [CH00].

The important remark of Curien and Herbelin was that for the following two tentative reductions:

$$c[\mu x.c'/\alpha] \overset{?}{\leftarrow} \langle \mu\alpha.c \,\|\, \mu x.c\rangle \overset{?}{\rightarrow} c'[\mu\alpha.c/x]$$

then the left one, giving priority to expressions, corresponds to call-by-value, denoting a strict reduction of the expression, while

$\mathsf{L}_{\mathbf{foc}}$ with $\hat{\mathsf{tp}}$
Syntax

pos. variables $x, y, z \ldots$
neg. variables $\alpha, \beta, \gamma \ldots$

(variables)	$\kappa ::= x \mid \alpha$
(values)	$V ::= V_+ \mid t_-$
(pos. values)	$V_+ ::= x \mid (V, V') \mid \{t_-\}$
(pos. terms)	$t_+ ::= x \mid (V, V') \mid \{t_-\} \mid \mu\alpha.c \mid \mu\hat{\mathsf{tp}}.c$
(neg. terms)	$t_- ::= \alpha \mid \mu(\kappa, \kappa').c \mid \mu\{\alpha\}.c \mid \mu x.c \mid \hat{\mathsf{tp}}$
(commands)	$c ::= \langle t_+ \parallel t_- \rangle \quad (=_\alpha \langle t_- \parallel t_+ \rangle)$

Reductions. Let $\to$ the contextual closure of the following rewrite rules:

$$(\mu\alpha) \qquad \langle \mu\alpha.c \parallel t_- \rangle \to_0 c[t_-/\alpha]$$
$$(\mu x) \qquad \langle V_+ \parallel \mu x.c \rangle \to_0 c[V_+/x]$$
$$(\mu(\kappa,\kappa')) \qquad \langle (V,V') \parallel \mu(\kappa,\kappa').c \rangle \to_0 c[V/\kappa, V'/\kappa']$$
$$(\mu\{\alpha\}) \qquad \langle \{t_-\} \parallel \mu\{\alpha\}.c \rangle \to_0 c[t_-/\alpha]$$
$$(\mu\hat{\mathsf{tp}})^* \qquad \left\langle \mu\hat{\mathsf{tp}}.\langle V_+ \parallel \hat{\mathsf{tp}} \rangle \,\middle\|\, t_- \right\rangle \to_0 \langle V_+ \parallel t_- \rangle$$

*with $\hat{\mathsf{tp}}$ possibly free in V_+
$\hat{\mathsf{tp}}$ is not subject to α conversion.

Equations. The following equations are contextual:

$$(\eta_+) \qquad \mu\alpha.\langle t_+ \parallel \alpha \rangle =_\eta t_+$$
$$(\eta_-)^* \qquad \mu q.\langle q \parallel t_- \rangle =_\eta t_-$$
$$(\mathsf{c}_V)^* \qquad \left\langle V \,\middle\|\, \mu q.\langle t_+ \parallel \mu q'.c \rangle \right\rangle =_\eta \left\langle t_+ \,\middle\|\, \mu q'.\langle V \parallel \mu q.c \rangle \right\rangle$$
$$(\eta_{\hat{\mathsf{tp}}}) \qquad \langle \mu\hat{\mathsf{tp}}.c \parallel \hat{\mathsf{tp}} \rangle =_\eta c$$
$$(\mathsf{c}_{\hat{\mathsf{tp}}})^* \qquad \left\langle \mu\hat{\mathsf{tp}}.\langle W \parallel \mu q.c \rangle \,\middle\|\, t_- \right\rangle =_\eta \left\langle W \,\middle\|\, \mu q.\langle \mu\hat{\mathsf{tp}}.c \parallel t_- \rangle \right\rangle$$

*where q is x, $\{\alpha\}$ or (κ, κ') and where W is V_+ or $\mu\hat{\mathsf{tp}}.c'$.

Figure 1: Syntax, reduction rules and equational theory of one-sided $\mathsf{L}_{\mathrm{foc}}$ *with* $\hat{\mathsf{tp}}$.

the right one, giving priority to contexts, corresponds to call-by-name (or the specific variant they studied), denoting a lazy reduction of the expression.[1]

1.1.2 Call-by-name, call-by-value...

Wadler showed with the Dual Calculus [Wad03] (along with an efficient introduction to symmetric syntaxes for sequent calculus) how it is possible in this kind of syntax to define abstraction and application in terms of simpler connectives. As introductory examples we show how we retrieve similar definitions in $\mathsf{L}_{\mathrm{foc},\hat{\mathsf{tp}}}$.

Note that the syntax in Fig. 1 allows for the two writings $\langle t_+ \parallel t_- \rangle$ and $\langle t_- \parallel t_+ \rangle$, which are considered equal in the same way as the λ calculus equates $\lambda x.x$ and $\lambda y.y$. Note also that reduction rules always give priority to positives t_+ over negatives t_-, in the sense that $\mu x.c$ only reduces when supplied a value V_+. This priority is further reflected in our convention that negative terms are values.

We will show how to derive both call-by-name and call-by-value abstractions and applications, and then we will explain what computational and logical meaning we give to an equality like $\langle t_+ \parallel t_- \rangle =_\alpha \langle t_- \parallel t_+ \rangle$.

The call-by-value implication can be derived if we choose to write $\langle t_+ \parallel t_- \rangle$ instead of $\langle t_- \parallel t_+ \rangle$ (that is, if we read t_+ as expressions).

Example 2 (Call-by-value implication)**.** Given t_+, u_+ positive terms of $\mathsf{L}_{\mathrm{foc},\hat{\mathsf{tp}}}$ and u_- a negative term, we define:

$$\langle \lambda x.t_+ \mid \stackrel{\text{def}}{=} \left\langle \{\mu(x,\alpha).\langle t_+ \parallel \alpha \rangle\} \,\middle|\right.$$
$$\left| t_+ \cdot u_- \right\rangle \stackrel{\text{def}}{=} \left| \mu\{\alpha\}.\left\langle t_+ \,\middle\|\, \mu x.\langle \alpha \parallel (x, u_-) \rangle \right\rangle \right\rangle$$
$$\langle t_+ u_+ \mid \stackrel{\text{def}}{=} \left\langle \mu\alpha.\langle t_+ \parallel u_+ \cdot \alpha \rangle \,\middle|\right. .$$

It is an easy exercise to check that these definitions allow us to retrieve the reduction rules described above for call-by-value.

Conversely, the call-by-name implication can be derived if we choose to write $\langle t_- \parallel t_+ \rangle$ instead of $\langle t_+ \parallel t_- \rangle$ (that is, if we read t_- as expressions).

[1] We believe that Danos, Joinet and Schellinx [DJS97] already had at least the intuition of this interpretation of the critical pair. However we could find no such formal claim in print.

Example 3 (Call-by-name implication)**.** Given t_-, u_- negative terms and V a positive value of $\mathsf{L}_{\mathrm{foc},\hat{\mathsf{tp}}}$, we define:

$$\langle \lambda\alpha.t_- \mid \stackrel{\text{def}}{=} \left\langle \mu(\alpha,x).\langle t_- \parallel x \rangle \right|$$
$$\left| t_- \cdot V \right\rangle \stackrel{\text{def}}{=} \left| (t_-, V) \right\rangle$$
$$\langle t_- u_- \mid \stackrel{\text{def}}{=} \left\langle \mu x.\langle t_- \parallel u_- \cdot x \rangle \right|$$

The reader might be surprised — here and later — to see the call-by-name abstraction written $\lambda\alpha$ rather than λx. This is due to the fact that we encode expressions with negative terms, and therefore our notation for negative variables (α, β...) apply, while the positive co-variables are written x, y... We found it necessary — with apologies to the reader — to make the difficult choice of going against tradition with our notations, because the distinction between polarities of variables is more important to stress in $\mathsf{L}_{\mathrm{foc}}$ than the distinction between variables and co-variables.

Similarly to the call-by-value case, and modulo the exchange of notations between α's and x's, one can easily check that these definitions allow us to retrieve the rules of reduction above for the Krivine machine (in particular stacks E become of the form $|\, V_+ \rangle$).

1.1.3 ... and polarisation

At this point the reader might think that our syntax is just an artificial way of writing the two deterministic variants of Curien-Herbelin $\tilde{\lambda}\mu\tilde{\mu}$ or Wadler's Dual Calculus with one syntax, based on a play on words (we leave the $\hat{\mathsf{tp}}$ part aside for the moment). This is only an illusion caused be our choice of presentation, as we explain now.

As made clear by Wadler [Wad03], the rules of negation in sequent calculus can be interpreted computationally as the constructors that exchange expressions and contexts — such constructors, written not, are unary, and they are such that if t is an expression, then $\mathsf{not}(t)$ is a context; and conversely.

In the Dual Calculus, $\mathsf{not}(t)$ is always a value even if t is not, and the constructs are given the following reduction rule:

$$\langle \mathsf{not}(t) \parallel \mathsf{not}(u) \rangle \to \langle u \parallel t \rangle$$

Positive and negative formulae

$$P ::= X \mid P_* \otimes P_* \mid \downarrow N_P$$
$$N ::= X^\perp \mid N_* \parr N_* \mid \uparrow P_P$$
$$P_* ::= P \mid N_P$$
$$N_* ::= N \mid P_P$$

Judgements

Γ, Δ of the form $\alpha : P_Q, x : N, \ldots$

T of the form $\hat{\mathsf{tp}} : P$

$$c : (\vdash \Gamma, T) \qquad \vdash V : P_*; \Gamma$$
$$\vdash t_- : N \mid \Gamma, T \qquad \vdash t_+ : P_Q \mid \Gamma, T$$

Implicit negation

$$(X)^\perp \stackrel{\text{def}}{=} X^\perp$$
$$(P_* \otimes Q_*)^\perp \stackrel{\text{def}}{=} {P_*}^\perp \parr {Q_*}^\perp$$
$$(\downarrow N_P)^\perp \stackrel{\text{def}}{=} \uparrow {N^\perp}_P$$

where ${N_P}^\perp$ is ${N^\perp}_P$ and $N^\perp$ is defined dually, such that $A^{\perp\perp} = A$.

Identity

$$\frac{}{\vdash x : P; x : P^\perp}\ (\mathrm{ax}_+) \qquad \frac{}{\vdash \alpha : N \mid \alpha : {N^\perp}_P, \hat{\mathsf{tp}} : P}\ (\mathrm{ax}_-)$$

$$\frac{c : (\vdash x : N, \Gamma, T)}{\vdash \mu x.c : N \mid \Gamma, T}\ (\mu_-) \qquad \frac{c : (\vdash \alpha : P_Q, \Gamma, T)}{\vdash \mu\alpha.c : P_Q \mid \Gamma, T}\ (\mu_+)$$

$$\frac{}{\vdash \hat{\mathsf{tp}} : N \mid \hat{\mathsf{tp}} : N^\perp}\ (\hat{\mathsf{tp}}) \qquad \frac{c : (\vdash \Gamma, \hat{\mathsf{tp}} : P)}{\vdash \mu\hat{\mathsf{tp}}.c : P_Q \mid \Gamma, \hat{\mathsf{tp}} : Q}\ (\mu\hat{\mathsf{tp}})$$

$$\frac{\vdash t_+ : P_Q \mid \Gamma, \hat{\mathsf{tp}} : R \qquad \vdash t_- : {P^\perp}_Q; \Delta}{\langle t_+ \parallel t_- \rangle : (\vdash \Gamma, \Delta, \hat{\mathsf{tp}} : R)}\ (\mathrm{cut})$$

Structure

$$\frac{\vdash V_+ : P; \Gamma}{\vdash V_+ : P_Q \mid \Gamma, \hat{\mathsf{tp}} : Q}\ (\mathrm{der}) \qquad \frac{\vdash t_- : N \mid \Gamma, \hat{\mathsf{tp}} : P}{\vdash t_- : N_P; \Gamma}\ (\mathrm{prom})$$

$$\frac{c : (\vdash \Gamma, T)}{c : (\vdash x : N, \Gamma, T)}\ (\mathrm{w}_+) \qquad \frac{c : (\vdash x : N, y : N, \Gamma, T)}{c\,[x/y] : (\vdash x : N, \Gamma, T)}\ (\mathrm{c}_+)$$

$$\frac{c : (\vdash \Gamma, T)}{c : (\vdash \alpha : P_Q, \Gamma, T)}\ (\mathrm{w}_-) \qquad \frac{c : (\vdash \alpha : P_Q, \beta : P_Q, \Gamma, T)}{c\,[\alpha/\beta] : (\vdash \alpha : P_Q, \Gamma, T)}\ (\mathrm{c}_-)$$

Plus rules similar to $(w_\pm)$ and $(c_\pm)$ for t_+, t_- and V which we do not mention.

Logic

$$\frac{\vdash V : P_*; \Gamma \qquad \vdash V' : Q_*; \Delta}{\vdash (V, V') : P_* \otimes Q_*; \Gamma, \Delta}\ (\otimes) \qquad \frac{c : (\vdash \kappa : N_*, \kappa' : M_*, \Gamma, T)}{\vdash \mu(\kappa, \kappa').c : N_* \parr M_* \mid \Gamma, T}\ (\parr)$$

$$\frac{\vdash t_- : N_P; \Gamma}{\vdash \{t_-\} : \downarrow N_P; \Gamma}\ (\downarrow) \qquad \frac{c : (\vdash \alpha : P_Q, \Gamma, T)}{\vdash \mu\{\alpha\}.c : \uparrow P_Q \mid \Gamma, T}\ (\uparrow)$$

Figure 2: A simple type-and-effect system for $\mathsf{L}_{\mathsf{foc},\hat{\mathsf{tp}}}$.

Thus if we go back to $\mathsf{L}_{\mathsf{foc},\hat{\mathsf{tp}}}$, we see that our equality:

$$\langle t_- \parallel t_+ \rangle =_\alpha \langle t_+ \parallel t_- \rangle$$

amounts to no less than omitting the constructors for negation – in the same way as negation is implicit in the proof nets of linear logic, for instance. As a consequence, negation becomes involutive, since, when not is unwritten, $\mathsf{not}(\mathsf{not}(t))$ is same as t.

Here we encounter the two "traps" mentioned by Girard [Gir91]. *Trap #1*: our α-equality above forces negation to be involutive; but it is false to think that forcing $\neg\neg A$ to be isomorphic to A was sufficient — just taking call-by-value or call-by-name classical logic and imposing this isomorphism as an additional axiom would not work, since this results in equating all terms (see e.g. [Gir91, Sel01]).

Trap #2: identifying $\langle t_+ \parallel t_- \rangle$ with $\langle t_- \parallel t_+ \rangle$ is not essential to our approach. In fact we could have made this involutive negation explicit in the syntax, as did Girard for LU [Gir93] or Danos et al. for $\mathsf{LK}^\eta_{\mathrm{pol}}$ [DJS97]. Their descriptions of the explicit involutive negation amount to the following: we observe that double negation is not involutive in the Dual Calculus because $\mathsf{not}(\mathsf{not}(t))$ (of type $\neg\neg A$) is a value even when t (of type A) is not — thus their computational behaviour can be different (for instance for $t = \mu\alpha.c$ with α fresh in c). The solution is to get rid of the restriction by which $\mathsf{not}(t)$ is a value, by allowing the computation to continue under not in case t is not a value.

We can do so by introducing rules like the following:

$$\forall t \text{ non-value}, \quad \langle \mathsf{not}(t) \parallel u \rangle \to_\varsigma \left\langle \mu x.\langle \mathsf{not}(x) \parallel u \rangle \,\middle\|\, t \right\rangle$$

(and symmetrically). On the left-hand side of this reduction, $\mathsf{not}(t)$ takes precedence whatever the u, while in the right-hand side, t takes precedence. Thus $\mathsf{not}(t)$ must be a positive expression, while t must be a positive context. We conclude that the connective of negation must exchange polarity of formulae (contrarily to the negations of e.g. the Dual Calculus), and only exists in a calculus where there are both positive and negative expressions.

We made this negation implicit instead of explicit in $\mathsf{L}_{\mathsf{foc}}$, for one reason: the CPS translation of LC is sort of an unsound semantics for the polarity-changing negation, in the sense that it anyway forces the identification of $\langle t_+ \parallel t_- \rangle$ with $\langle t_- \parallel t_+ \rangle$ (see Section 2.1). Therefore our analysis would be no better informed if we had explicit rules for negation: these rules would simply be mapped to the identity transformation on CPS terms.

1.1.4 Polarised connectives

On the logical side, the want for symmetry and in particular an involutive negation is certainly of interest. But here, we are also interested in polarisation because it allows us to mix positive and negative expressions.

The calculus we present has constructors for the *multiplicative* connectives of polarised classical logic, which are written $\otimes$ (the positive conjunction) and $\parr$ (the negative disjunction) which can be understood as a strict pair and a primitive and symmetric version of implication. Additives (the lazy pair & and the strict sum $\oplus$) as well as units are omitted since they are not useful for decomposing CPS translations.

$\mathsf{L}_{\mathsf{foc},\hat{\mathsf{tp}}}$ has a negation which we chose to leave implicit in the syntax, as we have already discussed. Also, it has coercions of polarities, or *shifts*, which can be implicit in the notation — for instance we can form strict pairs with negative terms. They can also be explicit using the "unary patterns" $\{\cdot\}$ and $\mu\{\cdot\}.c$[2], associated to the connectives $\downarrow$ and $\uparrow$. Explicit shifts are typically used at the root of a term, such as in the definition of call-by-value λ abstraction above. We could simulate explicit shifts by implicit ones, but our particular design choice will provide us accuracy in the decomposition of CPS translations, by allowing us to treat particularly such cases where the shift is at the root. Now, if we go back to the definitions of λ abstraction, we see that $\cdot^\perp \parr \cdot$ is indeed a primitive version of implication if we define the follow-

[2] We would have preferred to write these constructs $(\cdot)$ and $\mu(\cdot).c$ since they really read best as unary versions of the pair $(\cdot,\cdot)$. For clarity reasons however, because single parentheses are commonly used for grouping, we chose the notation $\{\cdot\}$ instead, with no strong preference.

ing:

$$\langle \lambda x.t_- \mid \stackrel{\text{def}}{=} \left\langle \mu(x,y).\langle t_- \parallel y\rangle \right|$$

From such a construct, we see that it is possible to derive the call-by-value (resp. call-by-name) λ abstraction defined above by only introducing the appropriate polarity coercions. (We compare such a primitive implication to a direct-style variant of Levy's Call-By-Push-Value paradigm [Lev99] which provides a similar decomposition of call-by-value and call-by-name implications in a non-symmetric setting.)

1.1.5 η restriction

On a more technical side, we note that positive terms enjoy what is called the η restriction in Danos *et al.* [DJS97], that is, any non-value is of the form $\mu\alpha.c$ or $\mu\hat{\mathsf{tp}}.c$, which is one way of expressing the focalisation of positive constructors. This corresponds also to the notion of *administrative normal form,* where (for instance) a pair of strict terms t and u (computed from left to right, say) is not given in the syntax but defined as:

$$(t,u) \stackrel{\text{def}}{=} \mathsf{let}\, x\, \mathsf{be}\, t\, \mathsf{in}\, \mathsf{let}\, y\, \mathsf{be}\, u\, \mathsf{in}\, (x,y)$$

which in L_{foc} is represented by $\mu\alpha.\left\langle t \,\middle\|\, \mu x.\langle u \parallel \mu y.\langle (x,y) \parallel e\rangle\right\rangle\right\rangle$.

This means that our diagrams omit (if we wanted to be consistent with traditions) one additional trivial step responsible for the restriction to administrative normal forms, starting from a variant of $\mathsf{L}_{\text{foc},\hat{\mathsf{tp}}}$ without this η restriction, where focalisation is expressed instead through reduction rules like the following:

$$\langle (t,u) \parallel e\rangle \to \left\langle t \,\middle\|\, \mu x.\langle u \parallel \mu y.\langle (x,y) \parallel e\rangle\right\rangle\right\rangle \quad \text{if } (t,u) \text{ is not a value}$$

The corresponding trivial translation step replaces (t,u) by its above definition, which reflects the arbitrary choice between left-to-right and right-to-left evaluation for the strict pair.

1.1.6 Delimited control operators

Following Ariola, Herbelin & Sabry [AHS04], we can derive *shift* and *reset* as follows:

$$\mathsf{S}(t_+) \stackrel{\text{def}}{=} \mu\alpha.\langle t_+\, \lambda x.\mu\hat{\mathsf{tp}}.\langle x \parallel \alpha\rangle \parallel \hat{\mathsf{tp}}\rangle$$
$$[t_+] \stackrel{\text{def}}{=} \mu\hat{\mathsf{tp}}.\langle t_+ \parallel \hat{\mathsf{tp}}\rangle$$

To help understand the rules associated to $\hat{\mathsf{tp}}$, let us introduce the following notation:

$$c\left[\hat{\mathsf{tp}} = t_1; \ldots; \hat{\mathsf{tp}} = t_n\right] \stackrel{\text{def}}{=} \left\langle \mu\hat{\mathsf{tp}}.\left\langle \cdots \langle \mu\hat{\mathsf{tp}}.c \parallel t_1\rangle \cdots \right\rangle \,\middle\|\, t_n \right\rangle$$

and let us write $\sigma = (\hat{\mathsf{tp}} = t_1; \ldots; \hat{\mathsf{tp}} = t_n)$ for such a thing looking like an environment. Note that the definition does not yield uniqueness of the decomposition of a command c under the form $c'[\sigma]$.

Now we can look at the rules of $\mathsf{L}_{\text{foc},\hat{\mathsf{tp}}}$, in particular the ones related to $\hat{\mathsf{tp}}$, as if they were rules of an environment-machine:

$$\langle \mu\hat{\mathsf{tp}}.c \parallel t_-\rangle[\sigma] = c[\hat{\mathsf{tp}} = t_-; \sigma]$$
$$\langle V_+ \parallel \hat{\mathsf{tp}}\rangle\left[\hat{\mathsf{tp}} = t_-; \sigma\right] \to \langle V_+ \parallel t_-\rangle[\sigma]$$
$$c\left[\hat{\mathsf{tp}} = \hat{\mathsf{tp}}; \sigma\right] =_\eta c[\sigma]$$
$$c\left[\hat{\mathsf{tp}} = \mu q.c'; \hat{\mathsf{tp}} = t_-; \sigma\right] =_\eta c\left[\hat{\mathsf{tp}} = \mu q.c'\left[\hat{\mathsf{tp}} = t_-\right]; \sigma\right]$$

The environment σ is only accessed when the variable $\hat{\mathsf{tp}}$ comes in head-position, that is to say, is encountered against a value V_+.

It is therefore sensible to consider a head-reduction strategy $\to_h \subseteq \to$ for terms of $\mathsf{L}_{\text{foc},\hat{\mathsf{tp}}}$ defined by the following:

$$c[\sigma] \to_h c'[\sigma] \text{ whenever } c \to_0 c' .$$

With this head-reduction we recover (an abstract and symmetric variant of) the environment machines for $\lambda\mu\hat{\mathsf{tp}}_\nu$ of Herbelin and Ghilezan [HG08]. (See also Herbelin's [Her05].)

1.2 CPS translation

See Fig. 3 for the syntax of the target of our CPS translations ($\lambda_\nu^\times$), and Fig. 4 for the CPS translation of $\mathsf{L}_{\text{foc},\hat{\mathsf{tp}}}$ into $\lambda_\nu^\times$.

The abstraction for a pair is obtained with $\lambda(x,y).t \stackrel{\text{def}}{=} \lambda z.\mathsf{let}\,(x,y)\,\mathsf{be}\,z\,\mathsf{in}\,t$.

Remark 4. In $\lambda_\nu^\times$, if $t \to_{\text{admin}} u$, then $t \simeq u$.

Proof. We note that we have in general:

$$\begin{aligned}(\lambda x.u)\,t &=_\eta (\lambda x.u)(\mathsf{let}\,x\,\mathsf{be}\,t\,\mathsf{in}\,x)\\ &=_\eta \mathsf{let}\,x\,\mathsf{be}\,t\,\mathsf{in}\,(\lambda x.u)\,x\\ &\to \mathsf{let}\,x\,\mathsf{be}\,t\,\mathsf{in}\,u\end{aligned}$$

Here we have $u = E^*[x]$ with E^* a finite sequence of linear contexts and therefore we can conclude (by induction on E^*) with a sequence of $=_\eta$ equivalences. □

Proposition 5 (Compositionality)**.** *For p a command or a term, x and α variables, V a value, V_+ a positive value and t_- a negative term, one has:*

$$\begin{aligned}\left(V\left[V_+/x\right]\right)_{\mathsf{VAL}} &= V_{\mathsf{VAL}}\left[V_{+\mathsf{VAL}}/x\right]\\ \left(p\left[V_+/x\right]\right)_{\mathsf{CPS}} &= p_{\mathsf{CPS}}\left[V_{+\mathsf{VAL}}/x\right]\\ \left(p\left[t_-/\alpha\right]\right)_{\mathsf{CPS}} &= p_{\mathsf{CPS}}\left[t_{-\mathsf{CPS}}/k_\alpha\right] .\end{aligned}$$

Proof. Here and in the remaining of the paper, proofs that are omitted are routine verification. □

Proposition 6 (Simulation)**.** *If $\mathsf{L}_{foc,\hat{\mathsf{tp}}} \vdash c \to c'$ then $\lambda_\nu^\times \vdash c_{\mathsf{CPS}} \to^+ {c'}_{\mathsf{CPS}}$.*

Proof. The result holds for $\to_0$ by case analysis. We conclude by induction on the definition of a contextual closure for $\to_0$. □

Proposition 7 (Soundness)**.** *If $\mathsf{L}_{foc,\hat{\mathsf{tp}}} \vdash c \simeq c'$ then $\lambda_\nu^\times \vdash c_{\mathsf{CPS}} \simeq {c'}_{\mathsf{CPS}}$. (Here and for other systems defined in this paper, we always define $\simeq \stackrel{def}{=} (=_\eta \cup \leftrightarrow)^*$.)*

Proof. Proof is routine; the key observation for the cases of the ($c_{\hat{\mathsf{tp}}}$) and (c_V) rules is that $(\lambda q.t)\,u \simeq \mathsf{let}\,q\,\mathsf{be}\,u\,\mathsf{in}\,t$ in $\lambda_\nu^\times$. □

1.3 A type-and-effect system

We propose in Fig. 2 a type system for $\mathsf{L}_{\text{foc},\hat{\mathsf{tp}}}$ which is an extension with *annotations of effects* of Girard's LC — or more precisely of Danos et al's $\mathsf{LK}^\eta_{\text{pol}}$ [DJS97].

In order to obtain a convenient type system, we had to add an intermediate judgement that corresponds to the *stoup*[3] of LC [Gir91]. This distinguished formula, which asserts a linearity constraint stronger than classical provability, is inspired by Andreoli's focusing proof search discipline [And92]. Its purpose in [Gir91] is to formulate the focalising cut elimination procedure. (See Curien & the author's [CM10] for a variant of L_{foc} where the role of the stoup is emphasised.)

Our stoup appears in the judgement $\vdash V : P_*; \Gamma$ where V is a value, possibly negative, and P_* is of the form P or N_P. We feel that allowing negatives in the stoup — a choice directly related to our convention that negatives are values — is a sensible choice for having a good treatment of coercions between polarities in our sequent calculus.[4]

[3] In fact it is slightly more constrained and corresponds more closely to Danos et al's η restriction.

[4] Tradition of having negatives in the stoup goes back to Girard [Gir93]. An alternative to our concise treatment would be to instantiate each logical rule for all possible combination of polarities of the premises, as in LC or LU [Gir91, Gir93]. (Witness the number of rules in these systems!)

Syntax

$$V ::= x \mid \lambda x.t \mid (V,V)$$
$$t ::= V \mid t\ t \mid \mathsf{let}\, x\, \mathsf{be}\, t\, \mathsf{in}\, t \mid \mathsf{let}\,(x,y)\,\mathsf{be}\, t\, \mathsf{in}\, t$$

Reduction rules

$$(\lambda x.t)\, V \to t\,[V/x]$$
$$\mathsf{let}\, x\, \mathsf{be}\, V\, \mathsf{in}\, t \to t\,[V/x]$$
$$\mathsf{let}\,(x,y)\,\mathsf{be}\,(V,V')\,\mathsf{in}\, t \to t[V/x,V'/y]$$

Equational theory Linear evaluation contexts are defined with:

$$E\Box ::= \mathsf{let}\, q\, \mathsf{be}\, \Box\, \mathsf{in}\, t \mid \mathsf{let}\, q\, \mathsf{be}\, t\, \mathsf{in}\, \Box \mid \Box\, t \mid t\, \Box$$

where q stands for x and (x,y).

$$\lambda x.(V\, x) =_\eta V$$
$$\mathsf{let}\, q\, \mathsf{be}\, t\, \mathsf{in}\, q =_\eta t$$
$$\mathsf{let}\, q\, \mathsf{be}\, t\, \mathsf{in}\, E[u] =_\eta E[\mathsf{let}\, q\, \mathsf{be}\, t\, \mathsf{in}\, u]$$
$$\mathsf{let}\, q\, \mathsf{be}\, V\, \mathsf{in}\, \lambda x.t =_\eta \lambda x.\mathsf{let}\, q\, \mathsf{be}\, V\, \mathsf{in}\, t$$

Administrative reduction We define for any $n \geq 0$,

$$(\lambda x.E_1[\cdots[E_n[x]]])\ t \to_{\mathrm{admin}} E_1[\cdots[E_n[t]]]$$

Figure 3: A λ_ν calculus with pairs: $\lambda_\nu^\times$.

VAL : $V \mapsto V$

$$x_{\mathsf{VAL}} \stackrel{\mathrm{def}}{=} x$$
$$(V,V')_{\mathsf{VAL}} \stackrel{\mathrm{def}}{=} (V_{\mathsf{VAL}}, V'_{\mathsf{VAL}})$$
$$\{t_-\}_{\mathsf{VAL}} = t_{-\mathsf{VAL}} \stackrel{\mathrm{def}}{=} t_{-\mathsf{CPS}}$$

CPS : $t_+ \mapsto t$

$$V_{+\mathsf{CPS}} \stackrel{\mathrm{def}}{=} \boldsymbol{k}\, V_{+\mathsf{VAL}}$$
$$(\mu\alpha.c)_{\mathsf{CPS}} \stackrel{\mathrm{def}}{=} \mathsf{let}\, k_\alpha\, \mathsf{be}\, \boldsymbol{k}\, \mathsf{in}\, c_{\mathsf{CPS}}$$
$$(\mu\hat{\mathsf{tp}}.c)_{\mathsf{CPS}} \stackrel{\mathrm{def}}{=} \boldsymbol{k}\, c_{\mathsf{CPS}}$$

CPS : $t_- \mapsto V$

$$\alpha_{\mathsf{CPS}} \stackrel{\mathrm{def}}{=} k_\alpha$$
$$\hat{\mathsf{tp}}_{\mathsf{CPS}} \stackrel{\mathrm{def}}{=} \lambda x.x$$
$$(\mu x.c)_{\mathsf{CPS}} \stackrel{\mathrm{def}}{=} \lambda x.c_{\mathsf{CPS}}$$
$$(\mu(\kappa,\kappa').c)_{\mathsf{CPS}} \stackrel{\mathrm{def}}{=} \lambda(\kappa,\kappa').c_{\mathsf{CPS}}$$
$$(\mu\{\alpha\}.c)_{\mathsf{CPS}} \stackrel{\mathrm{def}}{=} \lambda k_\alpha.c_{\mathsf{CPS}}$$

CPS : $c \mapsto t$

$$\langle t_+ \,\|\, u_-\rangle_{\mathsf{CPS}} \left(= \langle u_- \,\|\, t_+\rangle_{\mathsf{CPS}}\right) \stackrel{\mathrm{def}}{=} t_{+\mathsf{CPS}}\left[u_{-\mathsf{CPS}}/\boldsymbol{k}\right]$$

(κ being x or α in the source, and x or k_α in the target.)

Figure 4: The CPS translation of $\mathsf{L}_{\mathrm{foc},\hat{\mathsf{tp}}}$ into $\lambda_\nu^\times$.

CPS translation We assume that $\lambda_\nu^\times$ is given a standard system of simple types with connectives $\to, \times$. What justifies our type system is that it is mapped through the CPS translation into the type system of $\lambda_\nu^\times$.

Positive and annotated negative types translate as follows:

$$X_{\mathsf{CPS}} \stackrel{\mathrm{def}}{=} X \qquad (\downarrow N_P)_{\mathsf{CPS}} \stackrel{\mathrm{def}}{=} (N^\perp)_{\mathsf{CPS}} \to P_{\mathsf{CPS}}$$
$$(P_* \otimes Q_*)_{\mathsf{CPS}} \stackrel{\mathrm{def}}{=} P_{*\mathsf{CPS}} \times Q_{*\mathsf{CPS}} \qquad (N_P)_{\mathsf{CPS}} \stackrel{\mathrm{def}}{=} (N^\perp)_{\mathsf{CPS}} \to P_{\mathsf{CPS}}$$

Judgements translate as follows:

$$\begin{aligned} c : (\vdash \Gamma, \hat{\mathsf{tp}} : P) &\rightsquigarrow \Gamma^\perp{}_{\mathsf{CPS}} \vdash c_{\mathsf{CPS}} : P_{\mathsf{CPS}} \\ \vdash V : P_* \mid \Gamma &\rightsquigarrow \Gamma^\perp{}_{\mathsf{CPS}} \vdash V_{\mathsf{CPS}} : P_{*\mathsf{CPS}} \\ \vdash t_- : N \mid \Gamma, \hat{\mathsf{tp}} : P &\rightsquigarrow \Gamma^\perp{}_{\mathsf{CPS}} \vdash t_{-\mathsf{CPS}} : N_{P\mathsf{CPS}} \\ \vdash t_+ : P_Q \mid \Gamma, \hat{\mathsf{tp}} : R &\rightsquigarrow \Gamma^\perp{}_{\mathsf{CPS}}, \boldsymbol{k} : P^\perp{}_{Q\mathsf{CPS}} \vdash t_{+\mathsf{CPS}} : R_{\mathsf{CPS}} \end{aligned}$$

with $\Gamma^\perp{}_{\mathsf{CPS}} = \left(\vec{x} : \vec{N^\perp}_{\mathsf{CPS}}, \vec{k_\alpha} : \vec{P^\perp}_{Q\mathsf{CPS}}\right)$ whenever $\Gamma = (\vec{x} : \vec{N}, \vec{\alpha} : \vec{P_Q})$.

The fact that the type system we gave for $\mathsf{L}_{\mathrm{foc},\hat{\mathsf{tp}}}$ maps into a system of simple types for $\lambda_\nu^\times$ could be checked directly, but it will become obvious once we decompose the translation in smaller steps through LLP in Section 2.

1.4 Call-by-value delimited control

Fig. 5 gives the syntax, equations and CPS semantics of Ariola, Herbelin and Sabry's $\lambda\mu\hat{\mathsf{tp}}_\nu$ calculus [AHS04] along the lines of Herbelin and Ghilezan [HG08].[5]

Proposition 8 (Soundness). *If $\lambda\mu\hat{\mathsf{tp}}_\nu \vdash p \simeq p'$ then $\lambda_\nu^\times \vdash p_{\mathsf{CPS}\nu} \simeq p'_{\mathsf{CPS}\nu}$ where p is a term or a command.*

[5] We differ from [HG08] in the treatment of administrative redexes. If we wanted to be closer, we would have defined $(t\,u)_{\mathsf{CPS}\nu} = t_{\mathsf{CPS}\nu}\,[(\lambda x.u_{\mathsf{CPS}\nu}\,[(\lambda y.x\,(y,\boldsymbol{k}))/\boldsymbol{k}])/\boldsymbol{k}]$ and $(\mu\alpha.c)_{\mathsf{CPS}\nu} = c_{\mathsf{CPS}\nu}\,[\boldsymbol{k}/k_\alpha]$. But then we lose the fact that the administrative variable $\boldsymbol{k}$ is bound linearly. Yet the second part of the decomposition (Section 2) prompted us to be careful to treat "administratively" only those redexes that bind variables in a linear manner.

We come close to having a simulation result as well, but do not, essentially because of the structural substitution defined in the figure.

The definitions in Example 2 canonically induce a translation $\cdot_{\mathsf{POS}}$ from terms and commands of $\lambda\mu\hat{\mathsf{tp}}_\nu$ to terms and commands of $\mathsf{L}_{\mathrm{foc},\hat{\mathsf{tp}}}$ (just take $[e]$ to $|e\rangle$, x to x, $\mu\alpha$ to $\mu\alpha$ and $\mu\hat{\mathsf{tp}}$ to $\mu\hat{\mathsf{tp}}$):

$$\lambda\mu\hat{\mathsf{tp}}_\nu \xrightarrow{\cdot_{\mathsf{POS}}} \mathsf{L}_{\mathrm{foc},\hat{\mathsf{tp}}}$$

Proposition 9 (Soundness). *If $\lambda\mu\hat{\mathsf{tp}}_\nu \vdash p \simeq p'$ then $\mathsf{L}_{foc,\hat{\mathsf{tp}}} \vdash p_{\mathsf{POS}} \simeq p'_{\mathsf{POS}}$ where p is a term or a command.*

Proof. Routine verification except for $\lambda x.(V\,x) \simeq V$. The latter follows from the fact that we have $\{\mu x.\langle V \,\|\, \mu\{\alpha\}.\langle\alpha \,\|\, x\rangle\rangle\} \simeq V$ as a consequence of the rule (c_V). □

Proposition 10 (Factoring). *For p a command or a term of $\lambda\mu\hat{\mathsf{tp}}_\nu$, we have $p_{\mathsf{POS}\,\mathsf{CPS}} = p_{\mathsf{CPS}\nu}$.*

1.5 Call-by-name delimited control

Fig. 6 defines the syntax and reduction rules of the $\lambda\mu\hat{\mathsf{tp}}_n$ calculus along the lines of Herbelin-Ghilezan's [HG08] as well as a CPS translation into $\lambda_\nu^\times$ based on the Lafont-Reus-Streicher CPS translation [LRS93].

The CPS translation is essentially[6] the one of Herbelin & Ghilezan. However we replaced Herbelin & Ghilezan's target calculus (a call-by-name λ calculus with pairs) by $\lambda_\nu^\times$. We could do so because the image of this CPS translation is indifferent to the evaluation strategy as far as reduction rules are concerned. This is witnessed by the fact that this alternate semantics validates the same rules of reductions.

However, with $\lambda_\nu^\times$ as a target, the translation no longer validates the equation $[\hat{\mathsf{tp}}]\,\mu_n\hat{\mathsf{tp}}.c =_\eta c$ given in [HG08], in the sense that we do not have $\left([\hat{\mathsf{tp}}]\,\mu_n\hat{\mathsf{tp}}.c\right)_{\mathsf{CPS}n} \simeq c_{\mathsf{CPS}n}$ in general. This

[6] We differ from [HG08] in that we applied an η expansion in the case of "$\mu_n\hat{\mathsf{tp}}.c$", which leaves the denotational semantics of Herbelin & Ghilezan unchanged for a target in call-by-name. This is possible because $c_{\mathsf{CPS}n}$ only reduces to functions.

$\lambda\mu\hat{\mathsf{tp}}_v$

Syntax

$$V ::= x \mid \lambda x.t$$
$$t ::= V \mid t\ t \mid \mu\hat{\mathsf{tp}}.c \mid \mu\alpha.c$$
$$e ::= \alpha \mid \hat{\mathsf{tp}}$$
$$c ::= [e]\,t$$
$$\simeq \stackrel{\text{def}}{=} (\hookleftarrow \cup =_\eta)^*$$

Reductions The following reductions are contextual:

$$(\lambda x.t)\,V \to t\,[V/x]$$
$$(\mu\alpha.c)\,t \to \mu\beta.c\,[[\beta](\Box\,t)/\alpha] \quad (\beta \text{ fresh})$$
$$V\,(\mu\alpha.c) \to \mu\beta.c\,[[\beta](V\,\Box)/\alpha] \quad (\beta \text{ fresh})$$
$$[e]\,\mu\alpha.c \to c\,[e/\alpha]$$
$$\mu\hat{\mathsf{tp}}.[\hat{\mathsf{tp}}]\,V \to V \quad \text{(even if } \hat{\mathsf{tp}} \text{ occurs in } V)$$

(where $c\,[[\beta]\,E[\,]/\alpha]$ is the *structural substitution*: "replace all occurrences of the form $[\alpha]\,t$ in c by $[\beta]\,E[t]$".)

Equational theory The following equations are contextual:

$$E_v\,\Box ::= \Box \mid E_v[\Box\,t] \mid E_v[V\,\Box]$$
$$[\hat{\mathsf{tp}}]\,\mu\hat{\mathsf{tp}}.c =_\eta c$$
$$\mu\hat{\mathsf{tp}}.[e]\,((\lambda x.t)\,(\mu\hat{\mathsf{tp}}.c)) =_\eta (\lambda x.\mu\hat{\mathsf{tp}}.[e]\,M)\,(\mu\hat{\mathsf{tp}}.c)$$
$$\mu\alpha.[e]\,((\lambda x.t)\,u) =_\eta (\lambda x.\mu\alpha.[e]\,t)\,u$$
$$\mu\alpha.[\alpha]\,t =_\eta t$$
$$\lambda x.(V\,x) =_\eta V$$
$$(\lambda x.E_v[x])\,t =_\eta E_v[t]\,.$$

CPS translation from $\lambda\mu\hat{\mathsf{tp}}_v$ to $\lambda_v^\times$

VALv : $V \mapsto V$

$$x_{\mathsf{VAL}v} \stackrel{\text{def}}{=} x$$
$$(\lambda x.t)_{\mathsf{VAL}v} \stackrel{\text{def}}{=} \lambda(x,k).t_{\mathsf{CPS}v}\,[k/\boldsymbol{k}]$$

CPSv : $t \mapsto t$

$$V_{\mathsf{CPS}v} \stackrel{\text{def}}{=} \boldsymbol{k}\,V_{\mathsf{VAL}v}$$
$$(t\,u)_{\mathsf{CPS}v} \stackrel{\text{def}}{=} \mathsf{let}\,k\,\mathsf{be}\,\boldsymbol{k}\,\mathsf{in}\,t_{\mathsf{CPS}v}\,[\lambda x.u_{\mathsf{CPS}v}\,[\lambda y.x\,(y,k)/\boldsymbol{k}]/\boldsymbol{k}]$$
$$(\mu\alpha.c)_{\mathsf{CPS}v} \stackrel{\text{def}}{=} \mathsf{let}\,k_\alpha\,\mathsf{be}\,\boldsymbol{k}\,\mathsf{in}\,c_{\mathsf{CPS}v}$$
$$(\mu\hat{\mathsf{tp}}.c)_{\mathsf{CPS}v} \stackrel{\text{def}}{=} \boldsymbol{k}\,c_{\mathsf{CPS}v}$$

CPSv : $c \mapsto t$

$$([\alpha]\,t)_{\mathsf{CPS}v} \stackrel{\text{def}}{=} t_{\mathsf{CPS}v}\,[k_\alpha/\boldsymbol{k}]$$
$$([\hat{\mathsf{tp}}]\,t)_{\mathsf{CPS}v} \stackrel{\text{def}}{=} t_{\mathsf{CPS}v}\,[\lambda x.x/\boldsymbol{k}]$$

Figure 5: The $\lambda\mu\hat{\mathsf{tp}}_v$ calculus.

$\lambda\mu\hat{\mathsf{tp}}_n$

Syntax of $\lambda\mu\hat{\mathsf{tp}}_n$

$$t ::= \alpha \mid \lambda\alpha.t \mid t\ t \mid \mu_n\hat{\mathsf{tp}}.c \mid \mu x.c$$
$$c ::= [x]\,t \mid [\hat{\mathsf{tp}}]\,t$$
$$\simeq \stackrel{\text{def}}{=} (\hookleftarrow \cup =_\eta)^*$$

Reductions

$$(\lambda\alpha.t)\,u \to t\,[u/\alpha]$$
$$(\mu x.c)\,t \to \mu y.c\,[[y](\Box\,t)/x] \quad (y \text{ fresh})$$
$$[y]\,\mu x.c \to c\,[y/x]$$
$$\mu_n\hat{\mathsf{tp}}.[\hat{\mathsf{tp}}]\,t \to t \quad \text{(even if } \hat{\mathsf{tp}} \text{ occurs in } t)$$

Equational theory

$$\mu x.[x]\,t =_\eta t$$
$$\lambda\alpha.(t\,\alpha) =_\eta t$$

CPS translation from $\lambda\mu\hat{\mathsf{tp}}_n$ to $\lambda_v^\times$

CPSn : $t \mapsto V$

$$\alpha_{\mathsf{CPS}n} \stackrel{\text{def}}{=} k_\alpha$$
$$(\lambda\alpha.t)_{\mathsf{CPS}n} \stackrel{\text{def}}{=} \lambda(k_\alpha,k).t_{\mathsf{CPS}n}\,k$$
$$(t\,u)_{\mathsf{CPS}n} \stackrel{\text{def}}{=} \lambda k.t_{\mathsf{CPS}n}\,(u_{\mathsf{CPS}n},k)$$
$$(\mu x.c)_{\mathsf{CPS}n} \stackrel{\text{def}}{=} \lambda x.c_{\mathsf{CPS}n}$$
$$(\mu_n\hat{\mathsf{tp}}.c)_{\mathsf{CPS}n} \stackrel{\text{def}}{=} \lambda x.c_{\mathsf{CPS}n}\,x$$

CPSn : $c \mapsto t$

$$([x]\,t)_{\mathsf{CPS}n} \stackrel{\text{def}}{=} t_{\mathsf{CPS}n}\,x$$
$$([\hat{\mathsf{tp}}]\,t)_{\mathsf{CPS}n} \stackrel{\text{def}}{=} t_{\mathsf{CPS}n}$$

Figure 6: The $\lambda\mu\hat{\mathsf{tp}}_n$ calculus.

equation is the consequence of a linearity restriction that the $\mu_n\hat{\mathsf{tp}}.c$ binder enforces on the use of $\mu\hat{\mathsf{tp}}.c$, and the equation is not validated here because the restriction is not present in the broader $\mathsf{L}_{\mathsf{foc},\hat{\mathsf{tp}}}$.

Our explanation for the fact that we do not consider the equation $[\hat{\mathsf{tp}}]\,\mu_n\hat{\mathsf{tp}}.c =_\eta c$ is that the $\lambda\mu\hat{\mathsf{tp}}_n$ calculus is not defined by its CPS semantics but arose as a particular way to solve critical pairs in a non-deterministic $\lambda\mu\hat{\mathsf{tp}}$ calculus. But such a syntactic characterisation, as we see, is not devoid of ambiguity if we start looking at extensions of the calculus. Indeed, it is possible to see $\mathsf{L}_{\mathsf{foc},\hat{\mathsf{tp}}}$ in particular as one extension of $\lambda\mu\hat{\mathsf{tp}}_n$ that invalidates the equation.

Proposition 11 (Soundness)**.** *If $\lambda\mu\hat{\mathsf{tp}}_n \vdash p \simeq p'$ then $\lambda_v^\times \vdash p_{\mathsf{CPS}n} \simeq p'_{\mathsf{CPS}n}$ where p is a term or a command.*

Because of the structural substitution again, and like [HG08], we come close to having a simulation result as well, but do not.

Given t_- a negative term and c a command of $\mathsf{L}_{\mathsf{foc},\hat{\mathsf{tp}}}$, we define:

$$\langle \mu_n\hat{\mathsf{tp}}.c \mid \stackrel{\text{def}}{=} \left\langle \mu x.\langle \mu\hat{\mathsf{tp}}.c \,\|\, \mu(\alpha).\langle \alpha \,\|\, x\rangle\right\rangle \right|$$
$$[\hat{\mathsf{tp}}]\,t_- \stackrel{\text{def}}{=} \langle (t_-) \,\|\, \hat{\mathsf{tp}}\rangle\,.$$

Together with the definitions of Example 3, these definitions canonically induce a translation $\cdot_{\mathsf{NEG}}$ from terms and commands of $\lambda\mu\hat{\mathsf{tp}}_n$ to terms and commands of $\mathsf{L}_{\mathsf{foc},\hat{\mathsf{tp}}}$ (taking α to α, μx to μx and $[x]$ to $|\,x\rangle$):

$$\lambda\mu\hat{\mathsf{tp}}_n \xrightarrow{\cdot_{\mathsf{NEG}}} \mathsf{L}_{\mathsf{foc},\hat{\mathsf{tp}}}$$

Proposition 12 (Soundness)**.** *If $\lambda\mu\hat{\mathsf{tp}}_n \vdash p \simeq p'$ then $\mathsf{L}_{\mathsf{foc},\hat{\mathsf{tp}}} \vdash p_{\mathsf{NEG}} \simeq {p'}_{\mathsf{NEG}}$ where p is a term or a command.*

Proposition 13 (Factoring)**.** *For p a command or a term of $\lambda\mu\hat{\mathsf{tp}}_n$, we have $p_{\mathsf{NEG}\,\mathsf{CPS}} \to^*_{admin} p_{\mathsf{CPS}n}$.*

Proof. Routine verification for the classical subsystem (without $\hat{\mathsf{tp}}$) where we would even have equality. Thus the property follows from:

$$([\hat{\mathsf{tp}}]\,t)_{\mathsf{NEG}\,\mathsf{CPS}} = (\lambda x.x)\,t_{\mathsf{NEG}\,\mathsf{CPS}} \to_{\text{admin}} t_{\mathsf{NEG}\,\mathsf{CPS}}$$
$$(\mu_n\hat{\mathsf{tp}}.c)_{\mathsf{NEG}\,\mathsf{CPS}} = \lambda x.\,((\lambda k.k\,x)\,c_{\mathsf{NEG}\,\mathsf{CPS}}) \to_{\text{admin}} \lambda x.c_{\mathsf{NEG}\,\mathsf{CPS}}\,x\,.$$

□

1.6 The decomposition of type-and-effect systems

Herbelin and Ghilezan give in [HG08] type-and-effect systems for $\lambda\mu\hat{\mathsf{tp}}_v$ and $\lambda\mu\hat{\mathsf{tp}}_n$. For $\lambda\mu\hat{\mathsf{tp}}_v$, sequents are of the form:

$$\vec{x} : \vec{A} \vdash t : B_T \mid \vec{\alpha} : \vec{C_U}; \hat{\mathsf{tp}} : D$$

where the connective for implication has two annotations: $A_T \to B_U$. Now the positive coding of implication yields for A, B, T, U positive formulae of $\mathsf{L}_{\mathrm{foc},\hat{\mathfrak{tp}}}$ the following positive definition for this connective:

$$A_T \to B_U \stackrel{\text{def}}{=} \downarrow(A^{\perp} \parr B_U)_T .$$

It is related to the positive translation [Lau02] through the translation into $\mathsf{LLP}^{!_P ?_P}$ described in next section. With this definition we find back the type system for $\lambda\mu\hat{\mathfrak{tp}}_v$ as a sub-part of our type system for $\mathsf{L}_{\mathrm{foc},\hat{\mathfrak{tp}}}$.

For $\lambda\mu\hat{\mathfrak{tp}}_n$, sequents are of the form:

$$\vec{\alpha} : \vec{A_\Sigma} \vdash t : B \mid \vec{x} : \vec{C}; \hat{\mathfrak{tp}} : \Xi$$

with $\Sigma, \Xi ::= \perp \mid N \cdot \Sigma$, and where the connective for implication has one annotation: $A_\Sigma \to B$. Our coding of $\lambda\mu\hat{\mathfrak{tp}}_n$ yields, for A, B, C denoting negative formulae, the following definition:

$$A_\Sigma \to B \stackrel{\text{def}}{=} A^{\perp}{}_\Sigma \parr B$$
$$\perp \stackrel{\text{def}}{=} 0$$
$$C \cdot \Sigma \stackrel{\text{def}}{=} \downarrow C_\Sigma$$

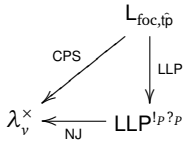
where 0 is the positive false constant. This allows us to retrieve the type-and-effect system of $\lambda\mu\hat{\mathfrak{tp}}_n$ as a sub-part. This coding is related to the negative translation of call-by-name $\lambda\mu$ into LLP [Lau02].

2 LLP with annotated exponentials

Laurent's LLP [Lau02, LR03] drew attention by giving a fresh view on translations of classical logic. LLP was provided with a theory of *proof nets*. Such a graphical representation helped in understanding that the (Lafont-Reus-Streicher variant [LRS93] of the) call-by-name CPS translation is very similar [Lau02, CM10] to Girard's translation of call-by-name λ calculus into linear logic, although the former could seem more intricate than the latter.

A key point for this clarification is that LLP decomposes the type of continuations into a co-variant modality "!" and an involutive negation.

We introduce a variant of Laurent's LLP we call $\mathsf{LLP}^{!_P ?_P}$. It factors the CPS translation of $\mathsf{L}_{\mathrm{foc},\hat{\mathfrak{tp}}}$ the following way:



where:

1. $\cdot_{\mathsf{LLP}}$ is an extension of the translation of LC into LLP given by Laurent [Lau02].

2. $\cdot_{\mathsf{NJ}}$, seen backwards, amounts to a rephrasing of the target of CPS translations, where we emphasise the duality between positives and negatives (in the sense of e.g. Zeilberger [Zei08, Zei09]).

See Fig. 7 for a definition of an untyped term syntax for $\mathsf{LLP}^{!_P ?_P}$ and Fig. 8 for a typing system. Like for $\mathsf{L}_{\mathrm{foc},\hat{\mathfrak{tp}}}$, the equation $\langle t_+ \,\|\, t_- \rangle =_\alpha \langle t_- \,\|\, t_+ \rangle$ is the consequence of our choice to let the involutive negation of LLP implicit.

We write $\mu(x,y)^\diamond.c \stackrel{\text{def}}{=} \mu z^\diamond.\langle z \,\|\, \mu(x,y).c \rangle$.

Proposition 14 (Confluence). *The reduction $\to$ is confluent.*

Proof. Again, the reduction $\to_0$ is defined by an orthogonal pattern rewrite system, which implies confluence for $\to$ (Nipkow [Nip91]). □

2.1 From $\mathsf{LLP}^{!_P ?_P}$ to $\lambda_v^\times$

In Fig. 9 we translate $\mathsf{LLP}^{!_P ?_P}$ into $\lambda_v^\times$. We observe the following:

1. The translation would identify $\langle t_+ \,\|\, t_- \rangle$ and $\langle t_- \,\|\, t_+ \rangle$ even if we provided $\mathsf{LLP}^{!_P ?_P}$ (and consequently $\mathsf{L}_{\mathrm{foc},\hat{\mathfrak{tp}}}$) with an explicit negation, which means that the CPS semantics only provides a coarse description of this negation;

2. A negative term t_- is sent onto a context of $\lambda_v^\times$, i.e. a term with a hole (written $\boldsymbol{k}$ here), with the type of t_- being mapped to the hole. As a consequence, the translation of sequent calculus into natural deduction accounts for the appearance of administrative redexes.

Proposition 15 (Simulation & soundness). *If $\mathsf{LLP}^{!_P ?_P} \vdash c \to c'$ then $\lambda_v^\times \vdash c_{\mathsf{NJ}} \to^i c'_{\mathsf{NJ}}$ for some $i \leq 1$. If $\mathsf{LLP}^{!_P ?_P} \vdash c =_\eta c'$ then $\lambda_v^\times \vdash c_{\mathsf{NJ}} =_\eta c'_{\mathsf{NJ}}$.*

Proposition 16 (Factoring). *For p a command or a term of $\mathsf{L}_{\mathrm{foc},\hat{\mathfrak{tp}}}$, we have $\lambda_v^\times \vdash p_{\mathsf{LLP}\,\mathsf{NJ}} =_\eta p_{\mathsf{CPS}}$ and for V a value we have $\lambda_v^\times \vdash V_{\mathsf{LLP}+\mathsf{NJ}} =_\eta V_{\mathsf{VAL}}$.*

Proof. We have:

$$(\mu\hat{\mathfrak{tp}}.c)_{\mathsf{LLP}\,\mathsf{NJ}} = \big(\mathsf{let}\, x\, \mathsf{be}\, \boldsymbol{k}\, \mathsf{in}\, \mathsf{let}\, y\, \mathsf{be}\, c_{\mathsf{LLP}\,\mathsf{NJ}}\, \mathsf{in}\, x\, y\big) =_\eta \boldsymbol{k}\, c_{\mathsf{LLP}\,\mathsf{NJ}} .$$

In all other cases we have an equality. □

Recall that $\lambda_v^\times$ is given a standard system of simple types with connectives $\to, \times$. Formulae and judgements translate as follows:

$$X_{\mathsf{NJ}} \stackrel{\text{def}}{=} X$$
$$(P \otimes Q)_{\mathsf{NJ}} \stackrel{\text{def}}{=} P_{\mathsf{NJ}} \times Q_{\mathsf{NJ}}$$
$$\big(!_p N\big)_{\mathsf{NJ}} \stackrel{\text{def}}{=} (N^{\perp})_{\mathsf{NJ}} \to P_{\mathsf{NJ}}$$
$$\vdash t : P \mid \Gamma \quad \rightsquigarrow \quad \Gamma^{\perp}{}_{\mathsf{NJ}} \vdash t_{\mathsf{NJ}} : P_{\mathsf{NJ}}$$
$$\vdash t : N \mid \Gamma, \star : P \quad \rightsquigarrow \quad \Gamma^{\perp}{}_{\mathsf{NJ}}, \boldsymbol{k} : N^{\perp}{}_{\mathsf{NJ}} \vdash t_{\mathsf{NJ}} : P_{\mathsf{NJ}}$$
$$c : (\vdash \Gamma, \star : P) \quad \rightsquigarrow \quad \Gamma^{\perp}{}_{\mathsf{NJ}} \vdash c_{\mathsf{NJ}} : P_{\mathsf{NJ}}$$

with $\Gamma^{\perp}{}_{\mathsf{NJ}} = \Big(\vec{x} : \vec{(N)^{\perp}}_{\mathsf{NJ}}\Big)$ whenever $\Gamma = \big(\vec{x} : \vec{N}\big)$.

2.2 From $\mathsf{L}_{\mathrm{foc},\hat{\mathfrak{tp}}}$ to $\mathsf{LLP}^{!_P ?_P}$

In Fig. 10 we translate $\mathsf{L}_{\mathrm{foc},\hat{\mathfrak{tp}}}$ into $\mathsf{LLP}^{!_P ?_P}$. This is an extension to delimited control of the translation of LC into LLP.

For non-delimited control and when negation is explicit, a sequent of LC is translated into a sequent of LLP according to the following pattern [Lau02]:

$$\vec{P}, \vec{N} \vdash_{\mathsf{LC}} \vec{Q}, \vec{M} \quad \rightsquigarrow \quad \vec{P}, !\vec{N} \vdash_{\mathsf{LLP}} ?\vec{Q}, \vec{M}$$

If we restrict to either polarity, this becomes:

$$\vec{P} \vdash_{\mathsf{LC}} \vec{Q} \quad \rightsquigarrow \quad \vec{P} \vdash_{\mathsf{LLP}} ?\vec{Q}$$
$$\vec{N} \vdash_{\mathsf{LC}} \vec{M} \quad \rightsquigarrow \quad !\vec{N} \vdash_{\mathsf{LLP}} \vec{M}$$

For the only-positive case, the construction is related to Moggi's description of computations in call-by-value through the Kleisli category for a monad, here the !? monad of LLP. The only-negative case, by symmetry, is related to the co-Kleisli of the ?! co-monad of LLP. (These two dual "mono-polarised" constructions have been investigated by Selinger [Sel01].)

The full translation thus describes a construction which is a generalisation of Kleisli constructions of both a monad and a co-monad.

Seen from the proof-theoretic side, the goal of this construction of LC from LLP is to conceal exponentials, such that shifts of

$\mathsf{LLP}^{!_P ?_P}$

Syntax uses positive variables $x, y \ldots$ and a distinguished negative variable $\star$.

(values)	$V ::= x \mid (V, V) \mid \mu x^{\diamond}.c$
(positive terms)	$t_+ ::= V \mid \mu\star.c$
(negative terms)	$t_- ::= \star \mid \mu x.c \mid \mu(x,x).c \mid V^{\diamond}$
(commands)	$c ::= \langle t_+ \parallel t_- \rangle \; (=_\alpha \langle t_- \parallel t_+ \rangle)$

Reductions Let $\to$ the contextual closure of the following rewrite rules:

$$\langle V \parallel \mu x.c \rangle \to_0 c\,[V/x]$$
$$\langle (V, V') \parallel \mu(x,y).c \rangle \to_0 c\,[V, V'/x, y]$$
$$\langle \mu x^{\diamond}.c \parallel V^{\diamond} \rangle \to_0 c\,[V/x]$$
$$\left\langle \mu\star.\langle t_+ \parallel \star \rangle \,\middle\|\, t_- \right\rangle \to_0 \langle t_+ \parallel t_- \rangle$$

Equational theory The following equations are contextual:

$$\mu q.\langle q \parallel t_- \rangle =_\eta t_- \qquad \left\langle t_+ \,\middle\|\, \mu q.\langle u_+ \parallel \mu q'.c \rangle \right\rangle =_\eta \left\langle u_+ \,\middle\|\, \mu q'.\langle t_+ \parallel \mu q.c \rangle \right\rangle$$
$$\mu x^{\diamond}.\langle V \parallel x^{\diamond} \rangle =_\eta V \qquad \left\langle t_+ \,\middle\|\, \mu q.\langle \mu\star.c \parallel u_- \rangle \right\rangle =_\eta \left\langle \mu\star.\langle t_+ \parallel \mu q.c \rangle \,\middle\|\, u_- \right\rangle$$
$$\langle \mu\star.c \parallel \star \rangle =_\eta c \qquad \left\langle V \,\middle\|\, \mu q.\langle \mu x^{\diamond}.c \parallel u_- \rangle \right\rangle =_\eta \left\langle \mu x^{\diamond}.\langle V \parallel \mu q.c \rangle \,\middle\|\, u_- \right\rangle$$

where q denotes x and (x, y).

Figure 7: Syntax, reduction rules and additional equations of $\mathsf{LLP}^{!_P ?_P}$.

Formulae

(positive)	$P ::= X \mid P \otimes P \mid !_P N$
(negative)	$N ::= X^{\perp} \mid N \parr N \mid ?_P P$

Implicit negation

$$(X)^{\perp} \stackrel{\text{def}}{=} X^{\perp} \qquad (P \otimes Q)^{\perp} \stackrel{\text{def}}{=} P^{\perp} \parr Q^{\perp} \qquad \left(!_P N\right)^{\perp} \stackrel{\text{def}}{=} ?_P N^{\perp}$$
$$(X^{\perp})^{\perp} \stackrel{\text{def}}{=} X \qquad (N \parr M)^{\perp} \stackrel{\text{def}}{=} N^{\perp} \otimes M^{\perp} \qquad \left(?_Q P\right)^{\perp} \stackrel{\text{def}}{=} !_Q P^{\perp}$$

Judgements

Γ, Δ of the form $x_1 : N_1, \ldots, x_n : N_n$
Π of the form $\star : P$ ($\Pi \neq \emptyset$ always)
$\vdash t_- : N \mid \Gamma, \Pi$
$\vdash t_+ : P \mid \Gamma$
$c : (\vdash \Gamma, \Pi)$

Identity

$$\frac{}{\vdash x : P \mid x : P^{\perp}}\ (\mathrm{ax}_+) \qquad \frac{}{\vdash \star : N \mid \star : N^{\perp}}\ (\mathrm{ax}_-)$$
$$\frac{c : (\vdash x : N, \Gamma)}{\vdash \mu x.c : N \mid \Gamma}\ (\mu_-) \qquad \frac{c : (\vdash \Gamma, \star : P)}{\vdash \mu\star.c : P \mid \Gamma}\ (\mu_+)$$
$$\frac{\vdash t_+ : P \mid \Gamma \qquad \vdash t_- : P^{\perp} \mid \Delta, \Pi}{\langle t_+ \parallel t_- \rangle : (\vdash \Gamma, \Delta, \Pi)}\ (\mathrm{cut})$$

Logic

$$\frac{\vdash V : P \mid \Gamma \qquad \vdash V' : Q \mid \Delta}{\vdash (V, V') : P \otimes Q \mid \Gamma, \Delta}\ (\otimes)$$
$$\frac{c : (\vdash x : N, y : M, \Gamma, \Pi)}{\vdash \mu(x,y).c : N \parr M \mid \Gamma, \Pi}\ (\parr)$$

Structure

$$\frac{c : (\vdash x : N, \Gamma, \star : P)}{\vdash \mu x^{\diamond}.c : !_P N \mid \Gamma}\ (\mathrm{prom}) \qquad \frac{\vdash V : P \mid \Gamma}{\vdash V^{\diamond} : ?_Q P \mid \Gamma, \star : Q}\ (\mathrm{der}) \qquad \frac{c : (\vdash \Gamma, \Pi)}{c : (\vdash x : N, \Gamma, \Pi)}\ (\mathrm{w}) \qquad \frac{c : (\vdash x : N, y : N, \Gamma, \Pi)}{c\,[x/y] : (\vdash x : N, \Gamma, \Pi)}\ (\mathrm{c})$$

Plus rules similar to (w) and (c) for t_+ and t_- we do not mention.

Figure 8: A system of simple types for $\mathsf{LLP}^{!_P ?_P}$.

$\mathsf{NJ} : t_+ \mapsto t$

$$x_{\mathsf{NJ}} \stackrel{\text{def}}{=} x$$
$$(V, V')_{\mathsf{NJ}} \stackrel{\text{def}}{=} (V_{\mathsf{NJ}}, V'_{\mathsf{NJ}})$$
$$(\mu x^{\diamond}.c)_{\mathsf{NJ}} \stackrel{\text{def}}{=} \lambda x.c_{\mathsf{NJ}}$$
$$(\mu\star.c)_{\mathsf{NJ}} \stackrel{\text{def}}{=} c_{\mathsf{NJ}}$$

$\mathsf{NJ} : t_- \mapsto t$

$$\star_{\mathsf{NJ}} \stackrel{\text{def}}{=} \boldsymbol{k}$$
$$(\mu x.c)_{\mathsf{NJ}} \stackrel{\text{def}}{=} \mathsf{let}\, x\, \mathsf{be}\, \boldsymbol{k}\, \mathsf{in}\, c_{\mathsf{NJ}}$$
$$(\mu(x,y).c)_{\mathsf{NJ}} \stackrel{\text{def}}{=} \mathsf{let}\,(x,y)\,\mathsf{be}\,\boldsymbol{k}\,\mathsf{in}\,c_{\mathsf{NJ}}$$
$$V^{\diamond}{}_{\mathsf{NJ}} \stackrel{\text{def}}{=} \boldsymbol{k}\, V_{\mathsf{NJ}}$$

$\mathsf{NJ} : c \mapsto t$

$$\langle t_+ \parallel u_- \rangle_{\mathsf{NJ}} \left(= \langle u_- \parallel t_+ \rangle_{\mathsf{NJ}}\right) \stackrel{\text{def}}{=} u_{-\mathsf{NJ}} \left[t_{+\mathsf{NJ}} / \boldsymbol{k}\right]$$

Figure 9: The translation of $\mathsf{LLP}^{!_P ?_P}$ into $\lambda_\nu^{\times}$

$$\underline{\textbf{LLP+} : V \mapsto V} \qquad \underline{\textbf{LLP} : t_+ \mapsto t_-} \qquad \underline{\textbf{LLP} : t_- \mapsto V}$$

$$x_{\mathsf{LLP+}} \stackrel{\text{def}}{=} x \qquad (V_+)_{\mathsf{LLP}} \stackrel{\text{def}}{=} \left((V_+)_{\mathsf{LLP+}}\right)^\circ \qquad \alpha_{\mathsf{LLP}} \stackrel{\text{def}}{=} k_\alpha$$

$$(V,V')_{\mathsf{LLP+}} \stackrel{\text{def}}{=} (V_{\mathsf{LLP+}}, V'_{\mathsf{LLP+}}) \qquad (\mu\alpha.c)_{\mathsf{LLP}} \stackrel{\text{def}}{=} \mu k_\alpha.c_{\mathsf{LLP}} \qquad \hat{\mathsf{tp}}_{\mathsf{LLP}} \stackrel{\text{def}}{=} \mu x^\circ.\langle x \parallel \star\rangle$$

$$\{t_-\}_{\mathsf{LLP+}} = t_{-\mathsf{LLP+}} \stackrel{\text{def}}{=} t_{-\mathsf{LLP}} \qquad (\mu\hat{\mathsf{tp}}.c)_{\mathsf{LLP}} \stackrel{\text{def}}{=} \mu x.\left\langle \mu\star.c_{\mathsf{LLP}} \,\middle\|\, \mu y.\langle y^\circ \parallel x\rangle \right\rangle \qquad (\mu x.c)_{\mathsf{LLP}} \stackrel{\text{def}}{=} \mu x^\circ.c_{\mathsf{LLP}}$$

$$\underline{\textbf{LLP} : c \mapsto c} \qquad (\mu(\kappa,\kappa').c)_{\mathsf{LLP}} \stackrel{\text{def}}{=} \mu(\kappa,\kappa')^\circ.c_{\mathsf{LLP}}$$

$$\langle t_+ \parallel t_-\rangle_{\mathsf{LLP}} \stackrel{\text{def}}{=} \langle t_{+\mathsf{LLP}} \parallel t_{-\mathsf{LLP}}\rangle \quad \left(\langle t_- \parallel t_+\rangle_{\mathsf{LLP}} \stackrel{\text{def}}{=} \langle t_{-\mathsf{LLP}} \parallel t_{+\mathsf{LLP}}\rangle\right) \qquad (\mu\{\alpha\}.c)_{\mathsf{LLP}} \stackrel{\text{def}}{=} \mu k_\alpha^\circ.c_{\mathsf{LLP}}$$

(κ being x or α in the source, and x or k_α in the target.)

Figure 10: The translation of $\mathsf{L}_{\mathsf{foc},\hat{\mathsf{tp}}}$ into $\mathsf{LLP}^{!_P ?_P}$.

polarities in LC can be transparent at the level of types, and unobtrusive at the level of terms. The least we can ask of a classical logic is indeed that there be no apparent modality in the system!

We reflect this in $\mathsf{L}_{\mathsf{foc},\hat{\mathsf{tp}}}$ and its type system: we provide implicit shifts, thanks to which it is possible to write the strict pair of a negative $t_- : N$ and a positive $V : P$ and obtain $(t_-, V) : N \otimes P$ for instance.An exponential is of course introduced in the translation into $\mathsf{LLP}^{!_P ?_P}$: $!N \otimes P$. But this exponential already belongs to the translation of $t_- : N$. We see that by construction, there is semantically no need in $\mathsf{L}_{\mathsf{foc},\hat{\mathsf{tp}}}$ to have a redundant symbol that would represent a shift in this place.

This involved two design choices: 1) dereliction is implicit because positive values and non-values share the same syntactical category; 2) promotion is implicit because we took the convention that negative terms are values.

Proposition 17 (Simulation). *If* $\mathsf{L}_{foc,\hat{\mathsf{tp}}} \vdash c \to c'$ *then* $\mathsf{LLP}^{!_P ?_P} \vdash c_{\mathsf{LLP}} \to^+ c'_{\mathsf{LLP}}$.

Proposition 18 (Soundness). *If* $\mathsf{L}_{foc,\hat{\mathsf{tp}}} \vdash c \simeq c'$ *then* $\mathsf{LLP}^{!_P ?_P} \vdash c_{\mathsf{LLP}} \simeq c'_{\mathsf{LLP}}$.

Types and sequents translate as follows:

$$X_{\mathsf{LLP}} \stackrel{\text{def}}{=} X \qquad (\downarrow N_P)_{\mathsf{LLP}} \stackrel{\text{def}}{=} !_{P_{\mathsf{LLP}}} N_{\mathsf{LLP}}$$

$$(P_* \otimes Q_*)_{\mathsf{LLP}} \stackrel{\text{def}}{=} P_{*\mathsf{LLP}} \otimes Q_{*\mathsf{LLP}} \qquad (N_P)_{\mathsf{LLP}} \stackrel{\text{def}}{=} !_{P_{\mathsf{LLP}}} N_{\mathsf{LLP}}$$

(and by duality ${P^\perp}_{\mathsf{LLP}} = {P_{\mathsf{LLP}}}^\perp$);

$$\begin{aligned}
(\vec{x} : \vec{N}, \vec{\alpha} : \vec{P_Q})_{\mathsf{LLP}} &\stackrel{\text{def}}{=} \left(\vec{x} : \vec{N}_{\mathsf{LLP}}, \vec{k_\alpha} : \vec{?_{Q_{\mathsf{LLP}}} P}_{\mathsf{LLP}}\right) \\
(\hat{\mathsf{tp}} : P)_{\mathsf{LLP}} &\stackrel{\text{def}}{=} (\star : P_{\mathsf{LLP}}) \\
c : (\vdash \Gamma, T) &\rightsquigarrow c_{\mathsf{LLP}} : (\vdash \Gamma_{\mathsf{LLP}}, T_{\mathsf{LLP}}) \\
\vdash t_- : N \mid \Gamma, \hat{\mathsf{tp}} : P &\rightsquigarrow \vdash t_{-\mathsf{LLP}} : !_{P_{\mathsf{LLP}}} N_{\mathsf{LLP}} \mid \Gamma_{\mathsf{LLP}} \\
\vdash V : P_* \mid \Gamma &\rightsquigarrow \vdash V_{\mathsf{LLP+}} : P_{*\mathsf{LLP}} \mid \Gamma_{\mathsf{LLP}} \\
\vdash t_+ : P_Q \mid \Gamma, T &\rightsquigarrow \vdash t_{+\mathsf{LLP}} : ?_{Q_{\mathsf{LLP}}} P_{\mathsf{LLP}} \mid \Gamma_{\mathsf{LLP}}, T_{\mathsf{LLP}} .
\end{aligned}$$

2.3 Translation into linear logic

In this section we explain the exponential $!_p$ through a translation of $\mathsf{LLP}^{!_P ?_P}$ into linear logic [Gir87] (LL). Because we miss a of standard acceptation of what would an untyped syntax of LL be (though we proposed one in [Mun09]), we restrict ourselves to the typed case (this also makes things simpler since we need to apply η expansion in the translation!).

Also, we are volutarily brief: the reader can already refer to Benton & Wadler [BW95] which establish a correpondence between the call-by-value calculus obtained by the boring translation and Moggi's monadic model for a commutative strong monad (recall that our $\lambda_\nu^\times$ is essentially Moggi's calculus plus equations reflecting commutativity of the monad).

In LL, it is possible to take the following definitions:

$$!_p A \stackrel{\text{def}}{=} !(A \parr P) \qquad ?_p A \stackrel{\text{def}}{=} ?(A \otimes P^\perp)$$

This modality enjoys the following rules:

$$\frac{\vdash ?\Gamma, A, P}{\vdash ?\Gamma, !_p A}\ (\text{prom}_P) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, ?_p A, P}\ (\text{der}_P)$$

The translation is and extension of Laurent and Regnier's *reversing* translation (ρ) of LLP into LL [LR03], which is itself inspired from Quatrini and Tortora de Falco [QTDF96].

The goal of ρ is to confine structural rules to places where they conform to the constraints of LL — thus it takes care that contraction, weakening and the context of promotion are restricted to formulae of the form $?A$.

At the level of types, all the translation requires is to replace all the atoms X and $X^\perp$ respectively by $!X$ and $?X^\perp$. At the level of terms, we can describe such a translation through η-expansion (*invertibility*) of negative connectives.

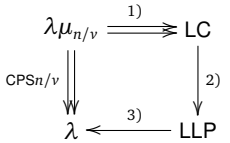
Since we are in a typed setting, η-expansion allows us to replace contractions and weakenings on formulae N by the same rules on formulae of the form $X^\perp$ or $?_Q P$. Indeed, for instance, if z is not linear in $c : (\vdash z : X^\perp \parr Y^\perp, \Gamma)$, then it is possible to get rid of structural rules on $X^\perp \parr Y^\perp$ by replacing c by $\langle \mu(x,y).c\,[(x,y)/z] \parallel z\rangle$, which is equivalent and only uses structural rules on $X^\perp$ and $Y^\perp$. The negative context of promotion is decomposed similarly so that it only contains formulae of the form $X^\perp$ or $?_Q P$ — in addition to the positive premise of our modified promotion, which is the main difference with the case of LLP. This promotion and dereliction are replaced by their derivation above.

We are done when we get rid of structural rules on atoms X and $X^\perp$ by replacing these atoms by $!X$ and $?X^\perp$ (the usual exponential, not the annotated one).

3 Comparison with other works

3.1 The polarised analysis of CPS translations

Our analysis of CPS translations remains valid if we restrict to non-delimited control: if we omit the $\hat{\mathsf{tp}}$ part from $\mathsf{L}_{\mathsf{foc},\hat{\mathsf{tp}}}$ we essentially get the syntax for Girard's polarised classical logic that we introduced in [Mun09]. A simple restriction of our syntax for $\mathsf{LLP}^{!_P ?_P}$ yields a syntax for LLP. Therefore, the introductory explanation of CPS translations is obtained by restricting $\lambda\mu\hat{\mathsf{tp}}_\nu$ and $\lambda\mu\hat{\mathsf{tp}}_n$ to the $\lambda\mu$ part. As we have seen in the details of the proofs, the factorings we obtain:



are even plain equalities — and our explanation is still relevant.

Many aspects of existing works are brought together by our account, as non-delimited CPS translations have already been extensively studied from the point of view of proof theory. We do not hope to establish a comprehensive list of contributions in this vast field, therefore we only mention the closest related works.

Beside Curien-Herbelin's $\bar{\lambda}\mu\tilde{\mu}$ [CH00], we inherit from Wadler's Dual Calculus [Wad03] the idea that abstraction and application can be derived even in a term syntax. There is no polarisation the works of Curien & Herbelin or Wadler, in the sense that call-by-value and call-by-name are conceived as global reduction strategies, and these calculi do not "mix" call-by-value and call-by-name in the same sense as $\mathsf{L}_{\mathrm{foc}}$.

Laurent introduced LLP and gave the "positive" translation into LLP for call-by-value $\lambda\mu$ calculus and the "negative" one for call-by-name $\lambda\mu$ calculus [Lau02]. Zeilberger later gave a logical and computational explanation of polarities in terms of values and computations. He described a system [Zei08] in which "*call-by-value and call-by-name evaluation freely interact*" in a "*programming language in which evaluation order is explicitly reflected at the level of types*", he claims. This allowed him to give a first explanation of CPS translations in terms of polarities [Zei09], based on Laurent's decomposition.

All these works themselves are part of a long series of works in proof theory or programming languages theory (sometimes both), including but not limited to ones of Girard, Danos et al., Laurent et al., Selinger, and others [And92, DJS97, Fil89, Gir87, Gir91, Gir93, Gri90, LRS93, LQdF05, Mog89, Mur92, Par92, QTDF96, Sel01].

While LLP and Zeilberger's system deal with polarities, they miss the classical polarities of LC and $\mathsf{LK}^{\eta}_{\mathrm{pol}}$. In particular, we incorrectly stated, at the time of [Mun09], that we shared with Zeilberger a computational interpretation of polarities. We should have looked more carefully, as we believe indeed that demand-driven and data-driven computations are an interpretation of the polarities of LC, and that the difference between these classical polarities and the notion of polarity in systems like LLP or Zeilberger's should not be overlooked.

For instance, Zeilberger described his positives as values while the positives of $\mathsf{L}_{\mathrm{foc}}$, like negatives, are computations that may carry effects. Also, the fact that LLP is based on the adjunction $? \dashv !$ (or equivalently $\uparrow \dashv \downarrow$ for Zeilberger's [Zei08]), but that the shifts in LC and $\mathsf{L}_{\mathrm{foc}}$ work the other way ($\uparrow$ is invertible while $\downarrow$ is not), makes a definitive distinction in how the two notions of polarity work.

By emphasizing the importance of the classical polarities of LC and $\mathsf{LK}^{\eta}_{\mathrm{pol}}$ in the decomposition, we believe that we brought one missing element to the big picture of how polarisation clarifies the contents of CPS translations.

3.2 Polarised linear logic and the study of double negation translations

Laurent and Regnier [Lau02, LR03] showed — among other contributions — that double-negation translations are not limited to the CPS translations that we can decompose in the spirit of the present paper.

More precisely, they decompose translations of classical logic into linear logic in two steps:

$$\lambda\mu \xrightarrow{G} \mathsf{LLP} \xrightarrow{K} \mathsf{LL}$$

where G represents the Girard-Laurent translations (with our definitions, $\cdot_{\mathsf{POS}\,\mathsf{LLP}}$ and $\cdot_{\mathsf{NEG}\,\mathsf{LLP}}$) and K can be the ρ translation or another translation described in [LR03] called *boxy*. They showed that these translations can be performed in the reverse order through a call-by-name λ calculus that they call CLC, to retrieve previous translations:

$$\lambda\mu \xrightarrow{K'} \mathsf{CLC} \xrightarrow{G'} \mathsf{LL}$$

where K' denote again double-negation translations. For instance, with G the negative translation (corresponding to the Lafont-Reus-Streicher one [LRS93]), and K the translation ρ, they show that G' is Girard's "$!A \multimap B$" translation and K' is nothing else than the Krivine double-negation translation!

We note that while some double-negation translations can be found in both the K' and the G steps, like the Lafont-Reus-Streicher translation, we have that Krivine's translation is specific to K', whereas translations that involve recursively defined values, like some CPS translations, are specific to G.

Our explanation of this fact is that the steps K and G answer distinct questions: the K translations answer to "*how do we obtain structural rules on negative formulae?*" whereas the G translations answer to "*how do we obtain structural rules on positive formulae?*". It is no surprise that in order to construct classical logic — which requires structural rules on all formulae — from linear logic, both questions have to be answered.

This delineates two families of double negation translations, and also means that it is false to identify double-negation translations in general with CPS translations.

3.3 Polarities and delimited control

The first attempt of applying polarisation to the study of delimited control goes back to Shan [Sha03]. Shan introduced an extension of LLP with linear shifts of polarities where he could translate Danvy-Filinski's calculus. He already hinted at a link with Girard's boring translation: his translation of intuitionistic logic into his system "*is essentially a polarised variant of Girard's 'boring' translation*", he writes. However this translation is more directly related to the CBV CPS translation that corresponds to the second stage of double CPS translations, than the boring translation. As a consequence, Shan's system seems designed to study double-CPS translations of delimited control, while we decompose single CPS translations. Shan does not study any other system with delimited control than Danvy-Filinski's.

Together with Kiselyov, Shan [KS07] later gave a type system and a type checking algorithm for Danvy-Filinski's calculus, whose goal is to be more liberal than the original type system of Danvy and Filinski, while we merely claim that (for the positive part) we retrieve the type system of $\lambda\mu\tilde{\mu}\hat{\mathrm{p}}_{v}$ [HG08]. Kiselyov and Shan use connectives $\cdot \downarrow T$ and $\cdot \uparrow T$, which correspond modulo duality to annotated negation modalities ($\neg_P Q$) similar to our exponentials $!_P \cdot$ and $?_P \cdot$. They note that typing a term amounts to transforming it into continuation-passing style. Our type system for $\mathsf{L}_{\mathrm{foc},\hat{\mathrm{p}}}$ does not improve this aspect, since it anticipates already to some degree the translation into $\mathsf{LLP}^{!_P ?_P}$, because we use explicit rules for getting in and out of the *stoup* that anticipate to some degree the translation into $\mathsf{LLP}^{!_P ?_P}$, but which are not reflected in the term syntax. However, "*investigating the delimited case*" of the duality between call-by-name and call-by-value classical logics was left for "*future work*" in Kiselyov & Shan [KS07].

More recently, Zeilberger proposed [Zei10] — among other contributions — a "*simple account of delimited continuations through a natural generalisation of classical polarised logic*" (not to be confused with polarised classical logic, i.e. LC or $\mathsf{LK}^{\eta}_{\mathrm{pol}}$) and "*in continuation-passing style*", thus expectedly located at the level of LLP. This work is based on hypotheses that make values returned by delimited continuations affect the structure of the polarity-changing negation.

Our own analysis of the polarised decomposition of CPS translations showed that the continuations are related to the rules for the shifts of LLP rather than to the polarity-changing negation, thanks to which we supposed instead that it is the structure of the shifts — here exponentials — that is affected.

In particular, the hypothesis that the symmetry between positive and negative connectives has to be "*broken*" does not seem necessary. Our own polarised target for delimited CPS translations ($\mathsf{LLP}^{!_P ?_P}$) contrasts indeed with Zeilberger's by being symmetric.[7]

Such a difference might find an explanation in the fact that polarisation is given a different meaning in the two works: inspired by a abstract description in the case of Zeilberger's [Zei10], and more faithful to linear logic in our case.

While it is expected that a formal translation of Danvy-Filinski's calculus be given into the positive part of Zeilberger's system, there are no claims of connection with other delimited control calculi of the literature.

3.4 The "subtractive" decomposition of delimited continuations

The work of Ariola, Herbelin and Sabry [AHS04] presents a study of delimited control calculi through translations into a calculus with a subtraction (see also [HGS10]). In linear logic, $?_P Q = ?(Q \otimes P^\perp)$ happens to look like a subtraction, and we expect that it is possible, as in Laurent-Regnier [LR03], to translate $\mathsf{L}_{\mathrm{foc},\hat{\mathrm{tp}}}$ into LL in the reverse order, starting with a reversing translation into a negative intuitionistic logic with a subtraction whose semantics would be inherited from LL's $?_P N^\perp$ modality. This might relate our decomposition to the one of Ariola *et al*, but would also yield translations in the style of Krivine that do not exist yet.

4 Perspectives

Our explanation of CPS translations shows the relevance of syntaxes à la $\bar{\lambda}\mu\tilde{\mu}$ / system L: their role is that of an interface between programming languages and semantics. On the one hand, they can be closer to the semantics than (idealised) programming languages, because they present computation in its interactive form (here, in interaction with some context). On the other hand, because we accept that the resulting system looks more like abstract machines, we have to devote the role of reflecting traditions of programming languages (such as presenting the features of the system in a functional style) to an additional layer of abstraction.

This two-step connection between programming languages and semantics is indeed what we find in our decomposition of CPS translations. Here, step 2) defines a CPS-like semantics. The interactive paradigm is relevant, because making explicit an interaction with some context is indeed the idea behind CPS translations. λ calculi with delimited control operators in call-by-value or call-by-name are then retrieved through the syntactic sugar of step 1), which answers the question "*How do we get a(n idealised) programming language back from a system such as* $\mathsf{L}_{\mathrm{foc},\hat{\mathrm{tp}}}$*?*". This question could merit a further development on its own, but is not directly related to denotational semantics.

Now this does not speak of the last step. In fact, step 3) seems to have little semantic value: on the contrary, this step necessarily equates commands *modulo duality*, and needless bureaucracy is introduced with administrative redexes, all of which seem to only serve the cause of obscurity of CPS translations. This suggests that $\mathsf{LLP}^{!_P ?_P}$ is a more appropriate target than the λ calculus if we want to understand, and be able to question, delimited CPS translations. (And we do not mean our particular syntax for $\mathsf{LLP}^{!_P ?_P}$, but more generally the system presented: 1- in sequent calculus style and 2- where the double negation modality is decomposed into two distinct co-variant modalities.)

But CPS translations are not the only source of semantics for which we found that system L provided relevant notation: in proof theory already, we found that $\mathsf{L}_{\mathrm{foc}}$ was the appropriate syntax with which we could define for Girard's LC a polarity-conscious variant of Krivine's classical realisability. (See [Mun09].)

As far as non-delimited CPS translations are concerned, a stronger result would be to prove the conjecture of Urban and Ratiu [UR06]. They suggest that regarding computation in classical logic, the tq-protocol of Danos-Joinet-Schellinx [DJS97] is as expressive as double-negation translations, by asking if the normal forms that can be obtained via *any* kind of double-negation translation could be obtained as well with the tq-protocol. So far, the result is only established for particular double-negation translations.

Urban and Ratiu mention two obstacles to the proof. One was the lack of an *usable* presentation of the tq-protocol. We believe that $\mathsf{L}_{\mathrm{foc}}$ provides an answer, being a rich syntax with a concise equational theory. (We can indeed simulate LK_{tq} in $\mathsf{LK}^{\eta}_{\mathrm{pol}}$, by encoding colours t and q with polarities, by introducing shifts for instance to simulate the lack of η restriction, and if necessary at the cost of replacing some additive connectives with multiplicatives and vice-versa.) The other obstacle is the absence of a positive definition of what a "double-negation translation" is. To this question we provide no answer.

If the one lesson we would retain from the call-by-value CPS translation is the construction by which strong monads describe call-by-value computation with effects, then the equivalent construction we should retain from our polarised decomposition of CPS translations is the one described in step 2).

This construction corresponds to a generalisation of monadic and co-monadic constructions; and therefore it describes how logical connectives and structures of computation that are algebraic can mix and interact with ones that are co-algebraic.

We believe that the question of a categorical account of step 2) is important for semantics of programming languages. Of course, "by duality", LC can be identified with its positive and negative subsystems, and we are back to the case of monads (e.g. [Sel01]). But we would request a description of the structure that treats the polarity-changing negation seriously, if only because the perfect symmetry induced by duality might not always be relevant, in particular for computer science. Twenty years after Girard wrote his landmark paper on polarisation [Gir91], this question still has no answer.

Acknowledgements

It was a real chance to benefit from the knowledge of Hugo Herbelin on delimited continuations during the many discussions that we had. I would also like to thank Beniamino Accattoli for discussions related to proof nets for $\mathsf{LLP}^{!_P ?_P}$. We did not mention such proof nets here, but they had a capital importance in the design of our decompositions and systems.

References

[AHS04] Zena M. Ariola, Hugo Herbelin, and Amr Sabry, *A type-theoretic foundation of continuations and prompts*, ICFP '04, ACM, 2004, pp. 40–53.

[7] For instance we are able to formally describe in $\mathsf{L}_{\mathrm{foc},\hat{\mathrm{tp}}}$ operators $\langle \mu\hat{\mathrm{tp}}.c |$ and $\langle \hat{\mathrm{tp}} |$: a dynamic *input* binder for a special *variable*. We did not mention them because we do not believe that structures obtained by duality are necessarily *useful*, but one might expect a completeness result for some co-monadic "effects" dual to the one of Filinski [Fil94].

[And92] Jean-Marc Andreoli, *Logic programming with focusing proof in linear logic*, Journal of Logic and Computation **2** (1992), no. 3, 297–347.

[BW95] Nick Benton and Philip Wadler, *Linear logic, monads, and the lambda calculus*, LICS '95, IEEE Computer Society Press, 1995, pp. 420–431.

[CH00] Pierre-Louis Curien and Hugo Herbelin, *The duality of computation*, ACM SIGPLAN Notices **35** (2000), 233–243.

[CM10] Pierre-Louis Curien and Guillaume Munch-Maccagnoni, *The duality of computation under focus*, Proc. IFIP TCS, 2010, Extended version.

[DF89] Olivier Danvy and Andrzej Filinski, *A functional abstraction of typed contexts*, Tech. report, 1989.

[DJS93] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx, *The structure of exponentials: Uncovering the dynamics of linear logic proofs*, Proc. KGC '93, Springer-Verlag, 1993.

[DJS97] ———, *A new deconstructive logic: Linear logic*, Journal of Symbolic Logic **62 (3)** (1997), 755–807.

[Fil89] Andrzej Filinski, *Declarative continuations and categorical duality*, Master's thesis, 1989.

[Fil94] ———, *Representing monads*, Proc. POPL, ACM Press, 1994, pp. 446–457.

[Gir87] Jean-Yves Girard, *Linear logic*, Theoretical Computer Science **50** (1987), 1–102.

[Gir91] ———, *A new constructive logic: Classical logic*, Math. Struct. Comp. Sci. (1991), no. 1.

[Gir93] ———, *On the unity of logic*, Ann. Pure Appl. Logic **59** (1993), no. 3, 201–217.

[Gri90] Timothy G. Griffin, *A formulae-as-types notion of control*, Seventeenth Annual ACM Symposium on Principles of Programming Languages, ACM Press, 1990, pp. 47–58.

[Her05] Hugo Herbelin, *C'est maintenant qu'on calcule, au cœur de la dualité*, 2005, Habilitation thesis.

[Her10] ———, *An intuitionistic logic that proves markov's principle*, LICS, 2010, pp. 50–56.

[HG08] Hugo Herbelin and Silvia Ghilezan, *An approach to call-by-name delimited continuations*, Proc. POPL '08, ACM, 2008.

[HGS10] Hugo Herbelin, Silvia Ghilezan, and Alexis Saurin, *A uniform approach to call-by-name and call-by-value delimited control*, Draft, January 2010.

[KH03] Yukiyoshi Kameyama and Masahito Hasegawa, *A sound and complete axiomatization of delimited continuations*, Proc. ICFP '03, ACM, 2003.

[KS07] Oleg Kiselyov and Chung-chieh Shan, *A substructural type system for delimited continuations*, Proc. TLCA'07, 2007.

[Lau02] Olivier Laurent, *Etude de la polarisation en logique*, Thèse de doctorat, Université Aix-Marseille II, mar 2002.

[Lev99] Paul Blain Levy, *Call-by-push-value: A subsuming paradigm*, Proc. TLCA '99, 1999, pp. 228–242.

[LQdF05] Olivier Laurent, Myriam Quatrini, and Lorenzo Tortora de Falco, *Polarized and focalized linear and classical proofs*, Ann. Pure Appl. Logic **134** (2005), no. 2-3, 217–264.

[LR03] Olivier Laurent and Laurent Regnier, *About translations of classical logic into polarized linear logic*, Proc. LICS '03, IEEE Computer Society, 2003.

[LRS93] Yves Lafont, B. Reus, and Thomas Streicher, *Continuation semantics or expressing implication by negation*, Tech. report, University of Munich, 1993.

[Mog89] Eugenio Moggi, *Computational lambda-calculus and monads*, IEEE Computer Society Press, 1989, pp. 14–23.

[MOTW94] John Maraist, Martin Odersky, David N. Turner, and Philip Wadler, *Call-by-name, call-by-value, call-by-need, and the linear lambda calculus*, Proc. MFPS '95, 1994.

[Mun09] Guillaume Munch-Maccagnoni, *Focalisation and classical realisability*, Proc. CSL '09, LNCS, Springer-Verlag, 2009.

[Mur92] Chetan R. Murthy, *A computational analysis of girard's translation and lc*, LICS, IEEE Computer Society, 1992, pp. 90–101.

[Nip91] Tobias Nipkow, *Higher-order critical pairs*, Proc. 6th IEEE Symp. Logic in Computer Science, IEEE Press, 1991, pp. 342–349.

[Par92] Michel Parigot, *Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction*, LPAR, 1992, pp. 190–201.

[Plo75] Gordon D. Plotkin, *Call-by-name, call-by-value and the lambda-calculus*, Theoretical Computer Science **1** (1975), no. 2, 125–159.

[QTDF96] Myriam Quatrini and Lorenzo Tortora De Falco, *Polarisation des preuves classiques et renversement*, Sciences, Paris t.322, Serie I, 1996.

[Sab96] Amr Sabry, *Note on axiomatizing the semantics of control operators*, Tech. Report CIS-TR-96-03, Department of Computer and Information Science, University of Oregon, 1996.

[Sel01] Peter Selinger, *Control categories and duality: On the categorical semantics of the lambda-mu calculus*, Math. Struct in Comp. Sci. **11** (2001), no. 2, 207–260.

[Sha03] Chung-chieh Shan, *From shift and reset to polarized linear logic*, Manuscript, 2003.

[Sha04] ———, *Delimited continuations in natural language: quantification and polarity sensitivity*, CW'04, 2004.

[UR06] Christian Urban and Diana Ratiu, *Classical logic is better than intuitionistic logic:a conjecture about double-negation translations*, CL&C '06, 2006.

[Wad94] Philip Wadler, *Monads and composable continuations*, Lisp Symb. Comput. **7** (1994), 39–56.

[Wad03] Philip Wadler, *Call-by-value is dual to call-by-name*, SIGPLAN Not. **38** (2003), no. 9, 189–201.

[Zei08] Noam Zeilberger, *On the unity of duality*, Ann. Pure and App. Logic **153:1** (2008).

[Zei09] ———, *The logical basis of evaluation order*, Ph.D. thesis, Carnegie Mellon University, 2009.

[Zei10] ———, *Polarity and the logic of delimited continuations*, Proceedings of LICS '10, 2010.