The duality of computation under focus

Pierre-Louis Curien, Guillaume Munch-Maccagnoni

The duality of computation under focus

Pierre-Louis Curien (CNRS, Paris 7, and INRIA) and Guillaume Munch-Maccagnoni (Paris 7 and INRIA)

Abstract. We review the close relationship between abstract machines for (call-by-name or call-by-value) λ-calculi (extended with Felleisen's $\mathcal{C}$) and sequent calculus, reintroducing on the way Curien-Herbelin's syntactic kit expressing the duality of computation. We use this kit to provide a term language for a presentation of LK (with conjunction, disjunction, and negation), and to transcribe cut elimination as (non confluent) rewriting. A key slogan here, which may appear here in print for the first time, is that commutative cut elimination rules are explicit substitution propagation rules. We then describe the focalised proof search discipline (in the classical setting), and narrow down the language and the rewriting rules to a confluent calculus (a variant of the second author's focalising system L). We then define a game of patterns and counterpatterns, leading us to a fully focalised finitary syntax for a synthetic presentation of classical logic, that provides a quotient on (focalised) proofs, abstracting out the order of decomposition of negative connectives.[1]

1 Introduction

This paper on one hand has an expository purpose and on the other hand pushes further the syntactic investigations on the duality of computation undertaken in [CH00].

Section 2 discusses the relation between familiar *abstract machines* for the λ-calculus (extended with control) and (classical) *sequent calculus*. Section 3 presents a faithful language (with a one-to-one correspondence between well-typed terms and proof trees) for a presentation of LK that anticipates a key ingredient of focalisation, by choosing a dissymetric presentation for the conjunction on one side and the disjunction on the other side of sequents. We recall the non-confluence of unconstrained classical cut-elimination.

In Section 4, we present the *focalised proof search discipline* (for *classical logic*), and adapt the syntactic framework of Section 3 to get a confluent system whose normal forms are precisely the terms denoting (cut-free) focalised proofs. The system we arrive at from these proof-search motivations is (a variant of) the second author's *focalising system* L (L_{foc}) [Mun09] We prove the completeness of L_{foc} with respect to LK for provability. In Section 5, we define some simple encodings having L_{foc} as source or target, indicating its suitability as an intermediate language (between languages and their execution or compilation).

Finally, in Section 6, we further reinforce the focalisation discipline, which leads us to *synthetic system* L ($\mathsf{L}_{\text{synth}}$), a logic of synthetic connectives in the spirit of Girard's ludics and Zeilberger's CU, for which we offer a syntactic account based on a simple game of patterns and counterpatterns that can be seen as another manifestation of dualities of computation. We show that the synthetic system L is complete with respect to focalising system L.

Notation. We shall write $t\{v/x\}$ the result of substituting v for x at all (free) occurrences of x in t, and $t[v/x]$ for an explicit operator [ACCL92] added to the language together with rules propagating it. Explicit substitutions are relevant here because they account for the commutative cut rules (see Section 3).

2 Abstract machines and sequent calculus

In this section, we would like to convey the idea that sequent calculus could have arisen from the goal of providing a typing system for the states of an abstract machine for the "mechanical evaluation of expressions" (to quote the title of Peter Landin's pioneering paper [Lan64]).

Here is a simple device for executing a (closed) λ-term in call-by-name (Krivine machine [Kri07]):

$$\boxed{\langle MN \mid E\rangle \longrightarrow \langle M \mid N \cdot E\rangle \qquad \langle \lambda x.M \mid N \cdot E\rangle \longrightarrow \langle M\{N/x\} \mid E\rangle}$$

[1] A slighlty shorter version appears in the Proceedings of the Conference IFIP TCS, Brisbane, Sept. 2010, published as a Springer LNCS volume., With respect to the published conference version, the present version corrects some minor mistakes in the last section, and develops a bit further the material of Section 5.

A state of the machine is thus a pair $\langle M \mid E \rangle$ where M is "where the computation is currently active", and E is the stack of things that are waiting to be done in the future, or the continuation, or the evaluation context. In λ-calculus litterature, contexts are more traditionally presented as terms with a hole: with this tradition, $\langle M \mid E \rangle$ (resp. $M \cdot E$) reads as $E[M]$ (resp. $E[[]M]$), or "fill the hole of E with M (resp. $[]M$)".

How can we type the components of this machine? We have three categories of terms and of typing judgements:

Expressions	Contexts	Commands
$M ::= x \mid \lambda x.M \mid MM$	$E ::= [\,] \mid M \cdot E$	$c ::= \langle M \mid E \rangle$
$(\Gamma \vdash M : A)$	$(\Gamma \mid E : A \vdash R)$	$c : (\Gamma \vdash R)$

where R is a (fixed) type of *final results*. The type of an expression (resp. a context) is the type of the value that it is producing (resp. expecting). The typing rules for contexts and commands are as follows:

$$\frac{}{\Gamma \mid [\,] : R \vdash R} \qquad \frac{\Gamma \vdash M : A \quad \Gamma \mid E : B \vdash R}{\Gamma \mid M \cdot E : A \to B \vdash R} \qquad \frac{\Gamma \vdash M : A \quad \Gamma \mid E : A \vdash R}{\langle M \mid E \rangle : (\Gamma \vdash R)}$$

and the typing rules for expressions are the usual ones for simply typed λ-calculus. Stripping up the term information, the second and third rules are rules of *sequent calculus* (left introduction of implication and cut).

We next review Griffin's typing of Felleisen's control operator $\mathcal{C}$. As a matter of fact, the behaviour of this constructor is best expressed at the level of an abstract machine:

$$\boxed{\langle \mathcal{C}(M) \mid E \rangle \longrightarrow \langle M \mid E^* \cdot [\,] \rangle \qquad \langle E^* \mid N \cdot E' \rangle \longrightarrow \langle N \mid E \rangle}$$

The first rule explains how the continuation E gets *captured*, and the second rule how it gets *restored*. Griffin [Gri90] observed that the typing constraints induced by the well-typing of these four commands are met when $\mathcal{C}(M)$ and E^* are typed as follows:

$$\frac{\Gamma \vdash M : (A \to R) \to R}{\Gamma \vdash \mathcal{C}(M) : A} \qquad \frac{\Gamma \mid E : A \vdash R}{\Gamma \vdash E^* : A \to R}$$

These are the rules that one adds to intutionistic natural deduction to make it classical, if we interpret R as $\bot$ (false), and if we encode $\neg A$ as $A \to R$. Hence, Griffin got no less than Curry-Howard for classical logic! But how does this sound in sequent calculus style? In classical sequent calculus, sequents have several formulas on the right and $\Gamma \vdash \Delta$ reads as "if all formulas in Γ hold, then at least one formula of Δ holds". Then it is natural to associate continuation variables with the formulas in Δ: a term will depend on its input variables, and on its output continuations. With this in mind, we can read the operational rule for $\mathcal{C}(M)$ as "$\mathcal{C}(M)$ is a map $E \mapsto \langle M \mid E^* \cdot [\,] \rangle$", and write it with a new binder (that comes from [Par92]):

$$\mathcal{C}(M) = \mu\beta.\langle M \mid \beta^* \cdot [\,] \rangle$$

where $[\,]$ is now a continuation variable (of "top-level" type R). Likewise, we synthesise $E^* = \lambda x.\mu\alpha.\langle x \mid E \rangle$, with α, x fresh, from the operational rules for E^* and for $\lambda x.M$.

The typing judgements are now: $(\Gamma \vdash M : A \mid \Delta)$, $(\Gamma \mid E : A \vdash \Delta)$, and $c : (\Gamma \vdash \Delta)$. The two relevant new typing rules are (axiom, right *activation*):

$$\frac{}{\Gamma \mid \alpha : A \vdash \alpha : A, \Delta} \qquad \frac{c : (\Gamma \vdash \alpha : A, \Delta)}{\Gamma \vdash \mu\alpha.c : A \mid \Delta}$$

plus a reduction rule: $\langle \mu\alpha.c \mid E \rangle \longrightarrow c\{E/\alpha\}$.

Note that in this setting, there is no more need to "reify" a context E into an expression E^*, as it can be directly substituted for a continuation variable.

Similarly, we can read off a (call-by-name) definition of MN from its operational rule: $MN = \mu\beta.\langle M \mid N.\beta \rangle$. Hence we can remove application from the syntax and arrive at a system in sequent calculus style *only* (no more elimination rule). This yields Herbelin's $\overline{\lambda}\mu$-calculus [Her95]:

$$\text{Expressions } M ::= x \mid \lambda x.M \mid \mu\alpha.c \qquad \text{Contexts } E ::= \alpha \mid M \cdot E \qquad \text{Commands } c ::= \langle M \mid E \rangle$$

which combines the first two milestones above: "sequent calculus", "classical".

Let us step back to the λ-calculus. The following describes a call-by-value version of Krivine machine:

$$\boxed{\langle MN \,|\, e\rangle \longrightarrow \langle N \,|\, M \odot e\rangle \qquad \langle V \,|\, M \odot e\rangle \longrightarrow \langle M \,|\, V \cdot e\rangle}$$

(the operational rule for $\lambda x.M$ is unchanged)[2]. Here, V is a *value*, defined as being either a variable or an abstraction (this goes back to [Plo75]). Again, we can read $M \odot e$ as "a map $V \mapsto \langle M \,|\, V \cdot e\rangle$", or, introducing a new binder $\tilde{\mu}$ (binding now ordinary variables):

$$M \odot e = \tilde{\mu}x.\langle M \,|\, x \cdot e\rangle$$

The typing rule for this operator is (left activation):

$$\frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \,|\, \tilde{\mu}x.c : A \vdash \Delta}$$

and the operational rule is $\langle V \,|\, \tilde{\mu}x.c\rangle \longrightarrow c\{V/x\}$ (V value).

Finally, we get from the rule for MN a call-by-value definition of application: $MN = \mu\alpha.\big\langle N \,\big|\, \tilde{\mu}x.\langle M \,|\, x \cdot \alpha\rangle\big\rangle$.

We have arrived at Curien and Herbelin's $\overline{\lambda}\mu\tilde{\mu}_Q$-calculus [CH00]:

Expressions $M ::= V^\diamond \mid \mu\alpha.c$	Values $V ::= x \mid \lambda x.M$	Contexts $e ::= \alpha \mid V \cdot e \mid \tilde{\mu}x.c$	Commands $c ::= \langle M \,|\, e\rangle$
$\Gamma \vdash M : A \,|\, \Delta$	$\Gamma \vdash V : A\,;\, \Delta$	$\Gamma \,|\, e : A \vdash \Delta$	$c : (\Gamma \vdash \Delta)$

with a new judgement for values (more on this later) and an explicit coercion from values to expressions. The syntax for contexts is both extended ($\tilde{\mu}x.c$) and restricted ($V \cdot e$ instead of $M \cdot e$). The reduction rules are as follows:

$$\boxed{\langle (\lambda x.M)^\diamond \,|\, V \cdot e\rangle \longrightarrow \langle M\{V/x\} \,|\, e\rangle \qquad \langle \mu\alpha.c \,|\, e\rangle \longrightarrow c\{e/\alpha\} \qquad \langle V^\diamond \,|\, \tilde{\mu}x.c\rangle \longrightarrow c\{V/x\}}$$

3 A language for LK proofs

In this section, we use some of the kit of the previous section to give a term language for classical sequent calculus LK, with negation, conjunction, and disjunction as connectives. Our term language is as follows:

$$\begin{array}{lll} \text{Commands} & c ::= \langle x \,|\, \alpha\rangle \mid \langle v \,|\, \alpha\rangle \mid \langle x \,|\, e\rangle \mid \langle \mu\alpha.c \,|\, \tilde{\mu}x.c\rangle \\ \text{Expressions} & v ::= (\tilde{\mu}x.c)^\bullet \mid (\mu\alpha.c, \mu\alpha.c) \mid inl(\mu\alpha.c) \mid inr(\mu\alpha.c) \\ \text{Contexts} & e ::= \tilde{\mu}\alpha^\bullet.c \mid \tilde{\mu}(x_1, x_2).c \mid \tilde{\mu}[inl(x_1).c_1 | inr(x_2).c_2] \end{array}$$

(In $\langle v \,|\, \alpha\rangle$ (resp. $\langle x \,|\, e\rangle$), we suppose α (resp. x) fresh for v (resp. e).) A *term* t is a command, an expression, or a context. As in section 2, we have three kinds of sequents: $(\Gamma \vdash \Delta)$, $(\Gamma \vdash A \,|\, \Delta)$, and $(\Gamma \,|\, A \vdash \Delta)$. We decorate LK's inference rules with terms, yielding the following typing system (one term construction for each rule of LK):

(axiom and cut/contraction) $$\frac{}{\langle x \,|\, \alpha\rangle : (\Gamma, x : A \vdash \alpha : A, \Delta)} \qquad \frac{c : (\Gamma \vdash \alpha : A, \Delta) \qquad d : (\Gamma, x : A \vdash \Delta)}{\langle \mu\alpha.c \,|\, \tilde{\mu}x.d\rangle : (\Gamma \vdash \Delta)}$$

(right) $$\frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \vdash (\tilde{\mu}x.c)^\bullet : \neg A \,|\, \Delta} \qquad \frac{c_1 : (\Gamma \vdash \alpha_1 : A_1, \Delta) \qquad c_2 : (\Gamma \vdash \alpha_2 : A_2, \Delta)}{\Gamma \vdash (\mu\alpha_1.c_1, \mu\alpha_2.c_2) : A_1 \wedge A_2 \,|\, \Delta}$$

$$\frac{c_1 : (\Gamma \vdash \alpha_1 : A_1, \Delta)}{\Gamma \vdash inl(\mu\alpha_1.c_1) : A_1 \vee A_2 \,|\, \Delta} \qquad \frac{c_2 : (\Gamma \vdash \alpha_2 : A_2, \Delta)}{\Gamma \vdash inr(\mu\alpha_2.c_2) : A_1 \vee A_2 \,|\, \Delta}$$

(left) $$\frac{c : (\Gamma \vdash \alpha : A, \Delta)}{\Gamma \,|\, \tilde{\mu}\alpha^\bullet.c : \neg A \vdash \Delta} \qquad \frac{c : (\Gamma, x_1 : A_1, x_2 : A_2 \vdash \Delta)}{\Gamma \,|\, \tilde{\mu}(x_1, x_2).c : A_1 \wedge A_2 \vdash \Delta} \qquad \frac{c_1 : (\Gamma, x_1 : A_1 \vdash \Delta) \qquad c_2 : (\Gamma, x_2 : A_2 \vdash \Delta)}{\Gamma \,|\, \tilde{\mu}[inl(x_1).c_1 | inr(x_2).c_2] : A_1 \vee A_2 \vdash \Delta}$$

(deactivation) $$\frac{\Gamma \vdash v : A \,|\, \Delta}{\langle v \,|\, \alpha\rangle : (\Gamma \vdash \alpha : A, \Delta)} \qquad \frac{\Gamma \,|\, e : A \vdash \Delta}{\langle x \,|\, e\rangle : (\Gamma, x : A \vdash \Delta)}$$

[2] The reason for switching notation from E to e will become clear in Section 5.

Note that the activation rules are packaged in the introduction rules and in the cut rule. As for the underlying sequent calculus rules, we have made the following choices:

1. We have preferred *additive* formulations for the cut rule and for the right introduction of conjunction (to stay in tune with the tradition of typed λ-calculi) over a multiplicative one where the three occurrences of Γ would be resp. Γ_1, Γ_2, and Γ_1, Γ_2 (idem for Δ). An important consequence of this choice is that contraction is a derived rule of our system, whence the name of *cut/contraction* rule above[3]:

$$\frac{\overline{\Gamma, A \vdash A, \Delta} \qquad \Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad\qquad \frac{\Gamma \vdash A, A, \Delta \qquad \overline{\Gamma, A \vdash A, \Delta}}{\Gamma \vdash A, \Delta}$$

2. Still in the λ-calculus tradition, weakening is "transparent". If $c : \Gamma \vdash \Delta$ is well-typed, then $c : (\Gamma, \Gamma' \vdash \Delta, \Delta')$ is well-typed (idem v, e). (Also, we recall that all free variables of c are among the ones declared in Γ, Δ.)
3. More importantly, we have adopted *irreversible* rules for right introduction of disjunction. On the other hand, we have given a *reversible* rule for left introduction of conjunction: the premise is derivable from the conclusion. *This choice prepares the ground for the next section on focalisation.*[4]

The relation between our typed terms and LK proofs is as follows.

- Every typing proof induces a proof tree of LK (one erases variables naming assumptions and conclusions, terms, the distinction between the three kinds of sequents, and the application of the deactivation rules).

- If bound variables are explicitly typed (which we shall refrain from doing in the sequel), then every provable typing judgement, say $\Gamma \mid e : A \vdash \Delta$, has a unique typing proof, i.e. all information is in Γ, A, Δ, e.

- If Π is an LK proof tree of $(A_1, \ldots, A_m \vdash B_1, \ldots, B_n)$, and if names $x_1, \ldots, x_m$, $\alpha_1, \ldots, \alpha_n$ are provided, then there exists a unique command $c : (x_1 : A_1, \ldots, x_m : A_m \vdash \alpha_1 : B_1, \ldots, \alpha_n : B_n)$, whose (unique) typing proof gives back Π by erasing.

With this syntax, we can express the cut-elimination rules of LK as *rewriting rules*:

Logical rules (redexes of the form $\langle \mu\alpha.\langle v \mid \alpha\rangle \mid \tilde{\mu}x.\langle x \mid e\rangle\rangle$):

$\langle \mu\alpha.\langle (\tilde{\mu}x.c)^\bullet \mid \alpha\rangle \mid \tilde{\mu}y.\langle y \mid \tilde{\mu}\alpha^\bullet.d\rangle\rangle \longrightarrow \langle \mu\alpha.d \mid \tilde{\mu}x.c\rangle$ (similar rules for conjunction and disjunction)

Commutative rules (going "up left", redexes of the form $\langle \mu\alpha.\langle v \mid \beta\rangle \mid \tilde{\mu}x.c\rangle$):

$\langle \mu\alpha.\langle (\tilde{\mu}y.c)^\bullet \mid \beta\rangle \mid \tilde{\mu}x.d\rangle \longrightarrow \left\langle \mu\beta'.\langle (\tilde{\mu}y.\langle \mu\alpha.c \mid \tilde{\mu}x.d\rangle)^\bullet \mid \beta'\rangle \,\middle|\, \tilde{\mu}y.\langle y \mid \beta\rangle \right\rangle$ ($\neg$ right)

(similar rules of commutation with the other right introduction rules and with the left introduction rules)

$\left\langle \mu\alpha.\langle \mu\beta.\langle y \mid \beta\rangle \mid \tilde{\mu}y'.c\rangle \,\middle|\, \tilde{\mu}x.d\right\rangle \longrightarrow \langle \mu\beta.\langle y \mid \beta\rangle \mid \tilde{\mu}y'.\langle \mu\alpha.c \mid \tilde{\mu}x.d\rangle\rangle$ (contraction right)

$\left\langle \mu\alpha.\langle \mu\beta'.c \mid \tilde{\mu}y.\langle y \mid \beta\rangle\rangle \,\middle|\, \tilde{\mu}x.d\right\rangle \longrightarrow \langle \mu\beta'.\langle \mu\alpha.c \mid \tilde{\mu}x.d\rangle \mid \tilde{\mu}y.\langle y \mid \beta\rangle\rangle$ (contraction left)

$\left\langle \mu\alpha.\langle \mu\alpha'.c \mid \tilde{\mu}x'.\langle x' \mid \alpha\rangle\rangle \,\middle|\, \tilde{\mu}x.d\right\rangle \longrightarrow \langle \mu\alpha.\langle \mu\alpha'.c \mid \tilde{\mu}x.d\rangle \mid \tilde{\mu}x.d\rangle$ (duplication)

$\langle \mu\alpha.\langle y \mid \beta\rangle \mid \tilde{\mu}x.d\rangle \longrightarrow \langle y \mid \beta\rangle$ (erasing)

Commutative rules (going "up right", redexes of the form $\langle \mu\alpha.c \mid \tilde{\mu}x.\langle y \mid e\rangle\rangle$): similar rules.

The (only?) merit of this syntax is its tight fit with proof trees and traditional cut elimination defined as transformations of undecorated proof trees. If we accept to losen this, we arrive at the following more "atomic" syntax:

Commands	$c ::= \langle v \mid e\rangle \mid c[\sigma]$
Expressions	$v ::= x \mid \mu\alpha.c \mid e^\bullet \mid (v, v) \mid inl(v) \mid inr(v) \mid v[\sigma]$
Contexts	$e ::= \alpha \mid \tilde{\mu}x.c \mid \tilde{\mu}\alpha^\bullet.c \mid \tilde{\mu}(x_1, x_2).c \mid \tilde{\mu}[inl(x_1).c_1 \mid inr(x_2).c_2] \mid e[\sigma]$

[3] In usual syntactic accounts of contraction, one says that if, say t denotes a proof of $\Gamma, x : A, y : A \vdash \Delta$, then $t[z/x, z/y]$ denotes a proof of $\Gamma, z : A \vdash \Delta$. Note that if this substitution is explicit, then we are back to an overloading of cut and contraction.

[4] For the same reason, we have chosen to take three connectives instead of just two, say, $\vee$ and $\neg$, because in the focalised setting $\neg(\neg A \vee \neg B)$ is only equivalent to $A \wedge B$ at the level of *provability*.

where σ is a list $v_1/x_1, \ldots, v_m/x_m, e_1/\alpha_1, \ldots, e_n/\alpha_n$. In this syntax, activation becomes "first class", and two versions of the axiom are now present (x, α, which give back the axiom of the previous syntax by deactivation). The typing rules are as follows (we omit the rules for $\tilde{\mu}x.c$, $\tilde{\mu}\alpha^\bullet.c$, $\tilde{\mu}(x_1, x_2).c$, $\tilde{\mu}[inl(x_1).c_1|inr(x_2).c_2]$, which are unchanged):

$$\frac{}{\Gamma\ ,\ x : A \vdash x : A \mid \Delta} \qquad \frac{}{\Gamma \mid \alpha : A \vdash \alpha : A\ ,\ \Delta} \qquad \frac{\Gamma \vdash v : A \mid \Delta \qquad \Gamma \mid e : A \vdash \Delta}{\langle v \mid e \rangle : (\Gamma \vdash \Delta)}$$

$$\frac{c : (\Gamma\ ,\ x : A \vdash \Delta)}{\Gamma \mid \tilde{\mu}x.c : A \vdash \Delta} \qquad \frac{c : (\Gamma \vdash \alpha : A\ ,\ \Delta)}{\Gamma \vdash \mu\alpha.c : A \mid \Delta}$$

$$\frac{\Gamma \mid e : A \vdash \Delta}{\Gamma \vdash e^\bullet : \neg A \mid \Delta} \qquad \frac{\Gamma \vdash v_1 : A_1 \mid \Delta \qquad \Gamma \vdash v_2 : A_2 \mid \Delta}{\Gamma \vdash (v_1, v_2) : A_1 \wedge A_2 \mid \Delta} \qquad \frac{\Gamma \vdash v_1 : A_1 \mid \Delta}{\Gamma \vdash inl(v_1) : A_1 \vee A_2 \mid \Delta} \qquad \frac{\Gamma \vdash v_2 : A_2 \mid \Delta}{\Gamma \vdash inr(v_2) : A_1 \vee A_2 \mid \Delta}$$

$$\frac{c : (\Gamma, x_1 : A_1, \ldots, x_m : A_m \vdash \alpha_1 : B_1, \ldots, \alpha_n : B_n) \ \ldots\ \Gamma \vdash v_i : A_i \mid \Delta \ \ldots\ \ldots\ \Gamma \mid e_j : B_j \vdash \Delta \ \ldots}{c[v_1/x_1, \ldots, v_m/x_m, e_1/\alpha_1, \ldots, e_n/\alpha_n] : (\Gamma \vdash \Delta)} \quad (\text{idem } v[\sigma], e[\sigma])$$

Note that we also have now *explicit substitutions* $t[\sigma]$, which feature a form of (multi-)cut where the receiver t's active formula, if any, is not among the cut formulas, in contrast with the construct $\langle v \mid e \rangle$ where the cut formula is active on both sides.

It is still the case that, by erasing, a well-typed term of this new syntax induces a proof of LK, and that all proofs of LK are reached (although not injectively anymore), since all terms of the previous syntax are terms of the new syntax. The rewriting rules divide now in *three* groups:

(control)	$\langle \mu\alpha.c \mid e \rangle \longrightarrow c[e/\alpha]$	$\langle v \mid \tilde{\mu}x.c \rangle \longrightarrow c[v/x]$
(logical)	$\langle e^\bullet \mid \tilde{\mu}\alpha^\bullet.c \rangle \longrightarrow c[e/\alpha]$	$\langle (v_1, v_2) \mid \tilde{\mu}(x_1, x_2).c \rangle \longrightarrow c[v_1/x_1, v_2/x_2]$
	$\langle inl(v_1) \mid \tilde{\mu}[inl(x_1).c_1 \mid inr(x_2).c_2] \rangle \longrightarrow c_1[v_1/x_1]$ (idem inr)	
(commutation)	$\langle v \mid e \rangle[\sigma] \longrightarrow \langle v[\sigma] \mid e[\sigma] \rangle$	
	$x[\sigma] \longrightarrow x$ (x not declared in σ)	$x[v/x, \sigma] \longrightarrow v$ (idem $\alpha[\sigma]$)
	$(\mu\alpha.c)[\sigma] \longrightarrow \mu\alpha.(c[\sigma])$ (idem $(\tilde{\mu}x.c)[\sigma]$) (capture avoiding)	
	(etc, no rule for composing substitutions)	

The *control rules* mark the decision to launch a substitution (and, in this section, of the direction in which to go, see below). The *logical rules* provide the interesting cases of cut elimination, corresponding to cuts where the active formula has been just introduced on both sides. The *commutative cuts* are now accounted for "trivially" by means of the *explicit substitution machinery* that carries substitution progressively inside terms towards their variable occurrences. Summarising, by liberalising the syntax, we have gained considerably in readability of the cut elimination rules[5].

Remark 1. In the "atomic" syntax, contractions are transcribed as terms of the form $\langle v \mid \beta \rangle$ where β occurs free in v, or of the form $\langle x \mid e \rangle$ where x occurs freely in e. If β (resp. x) does not occur free in v (resp. e), then the command expresses a simple deactivation.

The problem with classical logic viewed as a computational system is its wild non confluence, as captured by Lafont's critical pair [GLF89, DJS97], for which the $\mu\tilde{\mu}$ kit offers a crisp formulation. For any c_1, c_2 both of type $(\Gamma \vdash \Delta)$, we have (with α, x fresh for c_1, c_2, respectively):

$$c_1 \quad {}^*\!\longleftarrow \quad \langle \mu\alpha.c_1 \mid \tilde{\mu}x.c_2 \rangle \quad \longrightarrow^* \quad c_2$$

So, all proofs are identified... *Focalisation*, discussed in the next section, will guide us to solve this dilemma.

[5] The precise relation with the previous rules is as follows: for all s_1, s_2 such that $s_1 \longrightarrow s_2$ in the first system, there exists s such that $s_1 \longrightarrow^* s^* \longleftarrow s_2$ in the new system, e.g., for ($\neg$ right) $\langle \mu\alpha.\langle (\tilde{\mu}y.c)^\bullet \mid \beta \rangle \mid e \rangle \longrightarrow^* \langle (\tilde{\mu}y.(c[e/\alpha]))^\bullet \mid \beta \rangle \ {}^*\!\longleftarrow \left\langle \mu\beta'.\langle (\tilde{\mu}y.\langle \mu\alpha.c \mid e \rangle)^\bullet \mid \beta' \rangle \,\middle|\, \tilde{\mu}y.\langle y \mid \beta \rangle \right\rangle$.

4 A syntax for focalised classical logic

In this section, we adapt the *focalisation discipline* (originally introduced by [And92] in the setting of linear logic) to LK. A focalised proof search alternates between right and left phases, as follows:

- *Left phase*: Decompose (copies of) formulas on the left, in any order. Every decomposition of a negation on the left feeds the right part of the sequent. At any moment, one can change the phase from left to right.

- *Right phase*: Choose a formula A on the right, and *hereditarily* decompose a copy of it in all branches of the proof search. This *focusing* in any branch can only end with an axiom (which ends the proof search in that branch), or with a decomposition of a negation, which prompts a phase change back to the left. Etc...

Note the irreversible (or *positive, active*) character of the whole right phase, by the choice of A, by the choice of the left or right summand of a disjunction. One takes the risk of not being able to eventually end a proof search branch with an axiom. In contrast, all the choices on the left are reversible (or *negative, passive*). This strategy is not only complete (see below), it also guides us to design a disciplined logic whose behaviour will not collapse all the proofs.

To account for right focalisation, we introduce a fourth kind of judgement and a fourth syntactic category of terms: the *values*, typed as $(\Gamma \vdash V : A\,;\,\Delta)$ (the zone between the turnstyle and the semicolon is called the *stoup*, after [Gir91]). We also make official the existence of two disjunctions (since the behaviours of the conjunction on the left and of the disjunction on the right are different) and two conjunctions, by renaming $\wedge, \vee, \neg$ as $\otimes, \oplus, \neg^+$, respectively. Of course, this choice of linear logic like notation is not fortuitous. Note however that the source of distinction is not based here on the use of resources like in the founding work on linear logic, which divides the line between *additive* and *multiplicative* connectives. In contrast, our motivating dividing line here is that between *irreversible* and *reversible* connectives, and hopefully this provides additional motivation for the two conjunctions and the two disjunctions. Our formulas are thus defined by the following syntax:

$$P ::= X \,\big|\, P \otimes P \,\big|\, P \oplus P \,\big|\, \neg^+ P$$

These formulas are called positive. We can define their De Morgan duals as follows:

$$\overline{P_1 \otimes P_2} = \overline{P_1} \parr \overline{P_2} \qquad \overline{P_1 \oplus P_2} = \overline{P_1} \,\&\, \overline{P_2} \qquad \overline{\neg^+ P} = \neg^- \overline{P}$$

These duals are *negative* formulas: $N ::= \overline{X} \,\big|\, N \parr N \,\big|\, N \,\&\, N \,\big|\, \neg^- N$. They restore the duality of connectives, and are implicit in the presentation that follows (think of P on the left as being a $\overline{P}$ in a unilateral sequent $\vdash \overline{\Gamma}, \Delta$).

We are now ready to give the syntax of our calculus, which is a variant of the one given by the second author in [Mun09][6].

Commands	$c ::= \langle v \mid e \rangle \,\big|\, c[\sigma]$
Expressions	$v ::= V^\Diamond \,\big|\, \mu\alpha.c \,\big|\, v[\sigma]$
Values	$V ::= x \,\big|\, (V, V) \,\big|\, inl(V) \,\big|\, inr(V) \,\big|\, e^\bullet \,\big|\, V[\sigma]$
Contexts	$e ::= \alpha \,\big|\, \tilde{\mu}x.c \,\big|\, \tilde{\mu}\alpha^\bullet.c \,\big|\, \tilde{\mu}(x_1, x_2).c \,\big|\, \tilde{\mu}[inl(x_1).c_1 \mid inr(x_2).c_2] \,\big|\, e[\sigma]$

The typing rules are given in Figure 1. Henceforth, we shall refer to the calculus of this section (syntax + rewriting rules) as L_{foc}, and to the typing system as LKQ (after [DJS97]). Here are examples of proof terms in LKQ.

Example 1. $(\vdash (\tilde{\mu}(x, \alpha^\bullet).\langle x^\Diamond \mid \alpha\rangle)^\bullet : \neg^+(P \otimes \neg^+ P)\,;)$, where $\tilde{\mu}(x, \alpha^\bullet).c$ is an abbreviation for $\tilde{\mu}(x, y).\langle y^\Diamond \mid \tilde{\mu}\alpha^\bullet.c\rangle$.
$\big\langle inr((\tilde{\mu}x.\langle inl(x)^\Diamond \mid \alpha\rangle)^\bullet)^\Diamond \,\big|\, \alpha \big\rangle : (\vdash \alpha : P \oplus \neg^+ P)$.
$(\,|\, \tilde{\mu}(x_2, x_1).\langle (x_1, x_2)^\Diamond \mid \alpha\rangle : P_2 \otimes P_1 \vdash \alpha : P_1 \otimes P_2)$.

Proposition 1. *If $\Gamma \vdash \Delta$ is provable in* LK*, then it is provable in* LKQ.

[6] The main differences with the system presented in [Mun09] is that we have here an explicit syntax of values, with an associated form of typing judgement, while focalisation is dealt with at the level of the reduction semantics in [Mun09] (see also Remark 3). Also, the present system is bilateral but limited to positive formulas on both sides, it thus corresponds to the positive subsystem of the bilateral version of L_{foc} as presented in [Mun09][long version, Appendix A].

Fig. 1. System LKQ

$$\frac{}{\Gamma,\ x : P \vdash x : P\,;\,\Delta} \qquad \frac{}{\Gamma \mid \alpha : P \vdash \alpha : P\,,\ \Delta} \qquad \frac{\Gamma \vdash v : P \mid \Delta \qquad \Gamma \mid e : P \vdash \Delta}{\langle v \mid e \rangle : (\Gamma \vdash \Delta)}$$

$$\frac{c : (\Gamma,\ x : P \vdash \Delta)}{\Gamma \mid \tilde{\mu}x.c : P \vdash \Delta} \qquad \frac{c : (\Gamma \vdash \alpha : P\,,\ \Delta)}{\Gamma \vdash \mu\alpha.c : P \mid \Delta} \qquad \frac{\Gamma \vdash V : P\,;\,\Delta}{\Gamma \vdash V^{\diamond} : P \mid \Delta}$$

$$\frac{\Gamma \mid e : P \vdash \Delta}{\Gamma \vdash e^{\bullet} : \neg^{+}P\,;\,\Delta} \qquad \frac{\Gamma \vdash V_1 : P_1\,;\,\Delta \qquad \Gamma \vdash V_2 : P_2\,;\,\Delta}{\Gamma \vdash (V_1, V_2) : P_1 \otimes P_2\,;\,\Delta} \qquad \frac{\Gamma \vdash V_1 : P_1\,;\,\Delta}{\Gamma \vdash inl(V_1) : P_1 \oplus P_2\,;\,\Delta} \qquad \frac{\Gamma \vdash V_2 : P_2\,;\,\Delta}{\Gamma \vdash inr(V_2) : P_1 \oplus P_2\,;\,\Delta}$$

$$\frac{c : (\Gamma \vdash \alpha : P, \Delta)}{\Gamma \mid \tilde{\mu}\alpha^{\bullet}.c : \neg^{+}P \vdash \Delta} \qquad \frac{c : (\Gamma, x_1 : P_1, x_2 : P_2 \vdash \Delta)}{\Gamma \mid \tilde{\mu}(x_1, x_2).c : P_1 \otimes P_2 \vdash \Delta} \qquad \frac{c_1 : (\Gamma, x_1 : P_1 \vdash \Delta) \qquad c_2 : (\Gamma, x_2 : P_2 \vdash \Delta)}{\Gamma \mid \tilde{\mu}[inl(x_1).c_1 | inr(x_2).c_2] : P_1 \oplus P_2 \vdash \Delta}$$

$$\frac{\ldots \quad \Gamma \vdash V : P\,;\,\Delta \quad \ldots \quad \Gamma \mid e : Q \vdash \Delta \quad \ldots \quad c : (\Gamma \ldots, q : P, \ldots \vdash \Delta, \ldots, \alpha : Q, \ldots)}{c[\ldots, V/q, \ldots, e/\alpha] : (\Gamma \vdash \Delta)} \qquad (\text{idem } v[\sigma], V[\sigma], e[\sigma])$$

PROOF. Since we have defined a syntax for LK proofs in section 3, all we have to do is to translate this syntax into the focalised one. All cases are obvious (only inserting the coercion from values to expressions where appropriate) except for the introduction of $\otimes$ and $\oplus$ on the right, for which we can define $inl(\mu\alpha_1.c_1)$ as

$$\Gamma \vdash \mu\alpha.\big\langle \mu\alpha_1.c_1 \,\big|\, \tilde{\mu}x_1.\langle (inl(x_1))^{\diamond} \mid \alpha \rangle \big\rangle : P_1 \oplus P_2 \mid \Delta \qquad (\text{idem } \textit{inr})$$

and $(\mu\alpha_1.c_1, \mu\alpha_2.c_2)$ as $(\Gamma \vdash \mu\alpha.\Big\langle \mu\alpha_2.c_2 \,\Big|\, \tilde{\mu}x_2.\big\langle \mu\alpha_1.c_1 \,\big|\, \tilde{\mu}x_1.\langle (x_1, x_2)^{\diamond} \mid \alpha \rangle \big\rangle \Big\rangle : P_1 \otimes P_2 \mid \Delta)$. □

We make two observations on the translation involved in the proof of Proposition 1.

Remark 2. The translation *introduces cuts*: in particular, a cut-free proof is translated to a proof with (lots of) cuts. It also *fixes an order of evaluation*: one should read the translation of right introduction as a protocol prescribing the evaluation of the second element of a pair and then of the first (the pair is thus in particular *strict*, as observed in [Mun09]) (see also [Zei08, Lev04]). An equally reasonable choice would have been to permute the two $\tilde{\mu}$s: that would have encoded a left-to-right order of evaluation. This non-determinism of the translation has been known ever since Girard's seminal work [Gir91].

Remark 3. The translation is not reduction-preserving, which is expected (since focalisation induces restrictions on the possible reductions), but it is not reduction-reflecting either, in the sense that new reductions are possible on the translated terms. Here is an example (where, say $\mu\_.c$ indicates a binding with a dummy (i.e., fresh) variable). The translation of $\langle (\mu\_.c_1, \mu\_.c_2) \mid \tilde{\mu}x.c_3 \rangle$ rewrites to (the translation of) c_2:

$$\Big\langle \mu\alpha.\big\langle \mu\_.c_2 \,\big|\, \tilde{\mu}x_2.\langle \mu\_.c_1 \mid \tilde{\mu}x_1.\langle (x_1, x_2)^{\diamond} \mid \alpha \rangle \big\rangle \Big\rangle \,\Big|\, \tilde{\mu}x.c_3 \Big\rangle \quad \longrightarrow^{*} \quad \langle \mu\alpha.c_2 \mid \tilde{\mu}x.c_3 \rangle \quad \longrightarrow^{*} \quad c_2$$

while the source term is blocked. If we wanted to cure this, we could turn Proposition 1's encodings into additional rewriting rules in the source language. We refrain to do so, since we were merely interested in the source syntax as a stepping stone for the focalised one, and we are content that on one hand the rewriting system of Section 3 was good enough to eliminate cuts, and that on the other hand the focalised system is complete with respect to provability. But we note that the same additional rules *do* appear in the *target* language (and are called ς-rules, after [Wad03]) in [Mun09]. This is because in the focalised syntax proposed in [Mun09] there is no restriction on the terms of the language, hence $(\mu\_.c_1, \mu\_.c_2)$ is a legal term.

We move on to cut elimination, which (cf. Section 3) is expressed by means of three sets of rewriting rules, given in Figure 2. Note that we now have only one way to reduce $\langle \mu\alpha.c_1 \mid \tilde{\mu}x.c_2 \rangle$ (no more critical pair). As already stressed in

Fig. 2. Cut eliminition in L_{foc}

(control)	$\langle \mu\alpha.c \,|\, e\rangle \longrightarrow c[e/\alpha]$	$\langle V^\diamond \,|\, \tilde{\mu}x.c\rangle \longrightarrow c[V/x]$
(logical)	$\langle (e^\bullet)^\diamond \,|\, \tilde{\mu}\alpha^\bullet.c\rangle \longrightarrow c[e/\alpha]$	$\langle (V_1, V_2)^\diamond \,|\, \tilde{\mu}(x_1, x_2).c\rangle \longrightarrow c[V_1/x_1, V_2/x_2]$
	$\langle inl(V_1)^\diamond \,|\, \tilde{\mu}[inl(x_1).c_1 \,|\, inr(x_2).c_2]\rangle \longrightarrow c_1[V_1/x_1]$	$\langle inr(V_2)^\diamond \,|\, \tilde{\mu}[inl(x_1).c_1 \,|\, inr(x_2).c_2]\rangle \longrightarrow c_2[V_2/x_2]$
(commutation)	$\langle v \,|\, e\rangle[\sigma] \longrightarrow \langle v[\sigma] \,|\, e[\sigma]\rangle$ etc ...	

Section 3), the commutation rules are the usual rules defining (capture-avoiding) substitution. The overall operational semantics features call-by-value by the fact that variables x receive values, and features also call-by-name (through symmetry, see the logic LKT in Section 5) by the fact that continuation variables α receive contexts.

The reduction system presented in Figure 2 is *confluent*, as it is an orthogonal system in the sense of higher-order rewriting systems (left-linear rules, no critical pairs) [Nip93].

Remark 4. About μ: we note that an expression $\mu\beta.c$ is used only in a command $\langle \mu\beta.c \,|\, e\rangle$, and in such a context it can be expressed as $\langle (e^\bullet)^\diamond \,|\, \tilde{\mu}\beta^\bullet.c\rangle$, which indeed reduces to $c[e/\beta]$. However, using such an encoding would mean to shift from a direct to an indirect style for terms of the form $\mu\alpha.c$.

Proposition 2. *Cut-elimination holds in* LKQ.

PROOF. This is an easy consequence of the following three properties:

1) *Subject reduction.* This is checked as usual rule by rule.

2) *Weak normalisation.* One first gets rid of the redexes $\langle \mu\alpha.c \,|\, e\rangle$ by reducing them all (no such redex is ever created by the other reduction rules). As usual, one measures cuts by the size of the cut formula, called the degree of the redex, and at each step of normalisation, one chooses a redex of maximal degree all of whose subredexes have striclty lower degree. We then package reductions by considering $\langle V^\diamond \,|\, \tilde{\mu}x.c\rangle \longrightarrow c\{\!\{V/x\}\!\}$ (idem for the logical rules) as a single step, where $c\{\!\{\sigma\}\!\}$ is an augmented (implicit) substitution, defined by induction as usually except for $\langle v \,|\, e\rangle$:

$$\begin{array}{l}
\langle x^\diamond \,|\, \tilde{\mu}\alpha^\bullet.c\rangle\{\!\{e^\bullet/x, \sigma\}\!\} = c\{\!\{e/\alpha, e^\bullet/x, \sigma\}\!\} \\
\langle x^\diamond \,|\, \tilde{\mu}(x_1, x_2).c\rangle\{\!\{(V_1, V_2)/x, \sigma\}\!\} = c\{\!\{V_1/x_1, V_2/x_2, (V_1, V_2)/x, \sigma\}\!\} \\
\langle x^\diamond \,|\, \tilde{\mu}[inl(x_1).c_1 \,|\, inr(x_2).c_2]\rangle\{\!\{inl(V_1)/x, \sigma\}\!\} = c\{\!\{V_1/x_1, inl(V_1)/x, \sigma\}\!\} \quad \text{(idem } \textit{inr}\text{)} \\
\langle v \,|\, e\rangle\{\!\{\sigma\}\!\} = \langle v\{\!\{\sigma\}\!\} \,|\, e\{\!\{\sigma\}\!\}\rangle \quad \text{otherwise}
\end{array}$$

This is clearly a well-founded definition, by induction on the term in which substitution is performed (whatever the substitution is). This new notion of reduction ensures the following property: is $t_1 \longrightarrow t_2$ is obtained by reducing R_1 in t_1 and if R_2 is a redex created in t_2 by this (packaged) reduction, then R_1 is of the form $\langle e^\bullet \,|\, \tilde{\mu}\alpha^\bullet.c\rangle$, where c contains a subterm $\langle V^\diamond \,|\, \alpha\rangle$, which becomes R_2 in t_2. The key property is then that the degree of the created redex (the size of some formula P) is strictly smaller than the degree of the creating one (the size of $\neg^+P$)[7]. The other useful property is that residuals of redexes preserve their degree. Then the argument is easily concluded by associating to each term as global measure the multiset of the degrees of its redexes. This measure strictly decreases at each step (for the multiset extension of the ordering on natural numbers).

3) *Characterisation of normal forms.* A command in normal form has one of the following shapes (all contractions):

$$\langle V^\diamond \,|\, \alpha\rangle \qquad \langle x^\diamond \,|\, \tilde{\mu}\alpha^\bullet.c\rangle \qquad \langle x^\diamond \,|\, \tilde{\mu}(x_1, x_2).c\rangle \qquad \langle x^\diamond \,|\, \tilde{\mu}[inl(x_1).c_1 \,|\, inr(x_2).c_2]\rangle \qquad \square$$

Corollary 1. *Every sequent* $\Gamma \vdash \Delta$ *that is provable in* LK *admits a (cut-free) proof respecting the focalised discipline.*

[7] If we had not packaged reduction, we would have had to deal with the creation of redexes, say by susbstitution of some V for x, where the substitution could have been launched by firing a redex of the same degree as the created one.

PROOF. Let π be a proof of $\Gamma \vdash \Delta$. By Proposition 1, π translates to a command $c : (\Gamma \vdash \Delta)$, which by Proposition 2 reduces to a term denoting a cut-free proof. The LK proof obtained by erasing meets the requirement[8]. □

Also, by confluence and weak normalisation, LKQ is computationally coherent: $(x : P, y : P \vdash x : P\,;\,)$ and $(x : P, y : P \vdash y : P\,;\,)$ are not provably equal, being normal forms.

Our syntactic choices in this paper have been guided by the phases of focalisation. Indeed, with our syntax, the focalised proof search cycle can be represented as follows (following a branch from the root):

$$\begin{array}{ll}
\text{(right phase)} & \langle V^\Diamond \mid \alpha\rangle : (\Gamma \vdash \alpha : P, \Delta) \rightsquigarrow_{-/+} \Gamma \vdash V : P\,;\, \alpha : P, \Delta \rightsquigarrow_{+}^{*} \Gamma \vdash (\tilde\mu x.c)^\bullet : \neg^+Q\,;\,\Delta \\
& \quad \rightsquigarrow_{+/-} \Gamma \mid \tilde\mu x.c : Q \vdash \Delta \rightsquigarrow_{-} c : (\Gamma, x : Q \vdash \Delta) \qquad \text{(idem other } \tilde\mu \text{ binders)} \\
\text{(left phase)} & \langle x^\Diamond \mid \tilde\mu\alpha^\bullet.c\rangle : (\Gamma, x : \neg^+P \vdash \Delta) \rightsquigarrow_{-}^{*} c : (\Gamma, x : \neg^+P \vdash \alpha : P, \Delta) \\
& \langle x^\Diamond \mid \tilde\mu(x_1, x_2).c\rangle : (\Gamma, x : P_1 \otimes P_2 \vdash \Delta) \rightsquigarrow_{-}^{*} c : (\Gamma, x_1 : P_1, x_2 : P_2, x : P_1 \otimes P_2 \vdash \Delta) \\
& \langle x^\Diamond \mid \tilde\mu[inl(x_1).c_1 | inr(x_2).c_2]\rangle : (\Gamma, x : P_1 \oplus P_2 \vdash \Delta) \rightsquigarrow_{-}^{*} c_1 : (\Gamma, x_1 : P_1, x : P_1 \oplus P_2 \vdash \Delta) \\
& \langle x^\Diamond \mid \tilde\mu[inl(x_1).c_1 | inr(x_2).c_2]\rangle : (\Gamma, x : P_1 \oplus P_2 \vdash \Delta) \rightsquigarrow_{-}^{*} c_2 : (\Gamma, x_2 : P_2, x : P_1 \oplus P_2 \vdash \Delta)
\end{array}$$

Note that values and commands correspond to positive and negative phases, respectively. The other two categories of terms act as intermediates.

We can also add η-equivalences (or expansion rules, when read from right to left) to the system, as follows (where all mentioned variables are fresh for the mentioned terms):

$$\begin{array}{ll}
\mu\alpha.\langle v \mid \alpha\rangle = v & \tilde\mu(x_1, x_2).\langle (x_1, x_2)^\Diamond \mid e\rangle = e \\
\tilde\mu x.\langle x^\Diamond \mid e\rangle = e & \tilde\mu[inl(x_1).\langle inl(x_1)^\Diamond \mid e\rangle | inr(x_2).\langle inr(x_2)^\Diamond \mid e\rangle] = e \\
& \tilde\mu\alpha^\bullet\langle (\alpha^\bullet)^\Diamond \mid e\rangle = e
\end{array}$$

The rules on the left column allow us to cancel a deactivation followed by an activation (the control rules do the job for the sequence in the reverse order), while the rules in the right column express the reversibility of the negative rules.

Example 2. We relate $(\neg^+P_1) \otimes (\neg^+P_2)$ and $\neg^+(P_1 \oplus P_2)$ (cf. the well-know isomorphism of linear logic, reading $\neg^+P$ as $!\overline{P}$). There exist

$$c_1 : (y : \neg^+(P_1 \oplus P_2) \vdash \alpha : \neg^+P_1 \otimes \neg^+P_2) \qquad c_2 : (x : \neg^+P_1 \otimes \neg^+P_2 \vdash \gamma : \neg^+(P_1 \oplus P_2))$$

such that, say[9] $\langle \mu\gamma.c_2 \mid \tilde\mu y.c_1\rangle$, reduces $\langle x^\Diamond \mid \alpha\rangle : (x : \neg^+P_1 \otimes \neg^+P_2 \vdash \alpha : \neg^+P_1 \otimes \neg^+P_2)$. We set

$$\begin{array}{ll}
V_1 = ((\tilde\mu y_1'.\langle inl(y_1')^\Diamond \mid \beta\rangle)^\bullet, (\tilde\mu y_2'.\langle inr(y_2')^\Diamond \mid \beta\rangle)^\bullet) & \vdash V_1 : \neg^+P_1 \otimes \neg^+P_2\,;\,\beta : P_1 \oplus P_2 \\
V_2 = (\tilde\mu[inl(y_1).\langle y_1^\Diamond \mid \alpha_1\rangle | inr(y_2).\langle y_2^\Diamond \mid \alpha_2\rangle])^\bullet & \vdash V_2 : \neg^+(P_1 \oplus P_2)\,;\,\alpha_1 : P_1, \alpha_2 : P_2
\end{array}$$

We take $c_1 = \left\langle y^\Diamond \,\middle|\, \tilde\mu\beta^\bullet.\langle V_1^\Diamond \mid \alpha\rangle \right\rangle$ and $c_2 = \left\langle x^\Diamond \,\middle|\, \tilde\mu(\alpha_1^\bullet, \alpha_2^\bullet).\langle V_2^\Diamond \mid \gamma\rangle \right\rangle$, where $\tilde\mu(\alpha_1^\bullet, \alpha_2^\bullet).c$ is defined as a shorthand for, say $\tilde\mu(x_1, x_2).\left\langle x_2^\Diamond \,\middle|\, \tilde\mu\alpha_2^\bullet.\langle x_1^\Diamond \mid \tilde\mu\alpha_1^\bullet.c\rangle \right\rangle$. We have:

$$\begin{array}{rl}
\langle \mu\gamma.c_2 \mid \tilde\mu y.c_1\rangle \longrightarrow^* & \left\langle x^\Diamond \,\middle|\, \tilde\mu(\alpha_1^\bullet, \alpha_2^\bullet).\left\langle ((\tilde\mu y_1'.\langle (y')_1^\Diamond \mid \alpha_1\rangle)^\bullet, (\tilde\mu y_2'.\langle (y')_2^\Diamond \mid \alpha_2\rangle^\bullet) \,\middle|\, \alpha \right\rangle \right\rangle \\
& = \left\langle x^\Diamond \,\middle|\, \tilde\mu(\alpha_1^\bullet, \alpha_2^\bullet).\langle (\alpha_1^\bullet, \alpha_2^\bullet) \mid \alpha\rangle \right\rangle \\
& = \langle x^\Diamond \mid \alpha\rangle
\end{array}$$

We end the section with a lemma that will be useful in Section 6.

Lemma 1. – *If $\Gamma, x : \neg^+P \mid e : Q \vdash \Delta$, then $\Gamma \mid e\{\alpha^\bullet/x\} : Q \vdash \alpha : P, \Delta$.*
– *If $\Gamma, x : P_1 \otimes P_2 \mid e : Q \vdash \Delta$, then $\Gamma, x_1 : P_1, x_2 : P_2 \mid e\{(x_1, x_2)/x\} : Q \vdash \Delta$.*
– *If $\Gamma, x : P_1 \oplus P_2 \mid e : Q \vdash \Delta$, then $\Gamma, x_1 : P_1 \mid e\{inl(x_1)/x\} : Q \vdash \Delta$ and $\Gamma, x_2 : P_2 \mid e\{inr(x_2)/x\} : Q \vdash \Delta$.*
(and similarly for c, V, v), where $t\{V/x\}$ (resp. $t\{e/\alpha\}$) denotes the usual substitution (cf. Section 1).

[8] This argument of focalisation via normalisation goes back to [Gir91] (see also [Lau04] for a detailed proof in the case of linear logic).

[9] In the λ-calculus a provable isomorphism is a pair $(x : A \vdash v : B)$, $(y : B \vdash w : A)$ such that $w\{v/y\}$ reduces to x (and conversely). Here, we express this substitution (where v, w are not values) as $\mu\alpha.\left\langle v \,\middle|\, \tilde\mu y.\langle w \mid \alpha\rangle \right\rangle$, and the reduction as $\left\langle v \,\middle|\, \tilde\mu y.\langle w \mid \alpha\rangle \right\rangle \longrightarrow^* \langle x^\Diamond \mid \alpha\rangle$.

5 Encodings

Encoding CBV $\lambda(\mu)$-calculus into LKQ. We are now in a position to hook up with the material of Section 2. We can encode the call-by-value λ-calculus, by defining the following derived CBV implication and terms:

$$P \to^v Q = \neg^+(P \otimes \neg^+ Q)$$
$$\lambda x.v = ((\tilde{\mu}(x, \alpha^\bullet).\langle v \,|\, \alpha\rangle)^\bullet)^\diamond \qquad v_1 v_2 = \mu\alpha.\big\langle v_2 \,\big|\, \tilde{\mu}x.\langle v_1 \,|\, ((x, \alpha^\bullet)^\diamond)^\blacklozenge\rangle\big\rangle$$

where $\tilde{\mu}(x, \alpha^\bullet).c$ is the abbreviation used in Example 1 and where $V^\blacklozenge$ stands for $\tilde{\mu}\alpha^\bullet.\langle V^\diamond \,|\, \alpha\rangle$. These definitions provide us with a translation, which extends to (call-by-value) $\lambda\mu$-calculus [OS97, Roc05], and factors though $\overline{\lambda}\mu\tilde{\mu}_Q$-calculus (cf. Section 2), defining $V \cdot e$ as $(V, e^\bullet)^\blacklozenge$[10]. The translation makes also sense in the untyped setting, as the following example shows.

Example 3. Let $\Delta = \lambda x.xx$. We have $[\![\Delta\Delta]\!]^+_{\mathrm{v}} = \mu\gamma.c$, and $c \longrightarrow^* c$, with

$$c = \big\langle (e^\bullet)^\diamond \,\big|\, \tilde{\mu}z.\langle (e^\bullet)^\diamond \,|\, (z, \gamma^\bullet)^\blacklozenge\rangle\big\rangle \quad \text{and} \quad e = \tilde{\mu}(x, \alpha^\bullet).\big\langle x^\diamond \,\big|\, \tilde{\mu}y.\langle x^\diamond \,|\, (y, \alpha^\bullet)^\blacklozenge\rangle\big\rangle$$

Encoding CBN $\lambda(\mu)$-calculus. What about CBN? We can translate it to LKQ, but at the price of translating terms to contexts, which is a violence to our goal of giving an intuitive semantics to the first abstract machine presented in Section 2. Instead, we spell out the system dual to LKQ, which is known as LKT, in which expressions and contexts will have negative types, and in which we shall be able to express CBN λ-terms as expressions. Our syntax for LKT is a mirror image of that for LKQ: it exchanges the μ and $\tilde{\mu}$, the x's and the α's, etc..., and renames *inl*, *inr* as *fst*, *snd*, which are naturally associated with $\&$ while the latter were naturally associated with $\oplus$:

$$\begin{array}{ll} \text{Commands} & c ::= \langle v \,|\, e\rangle \\ \text{Covalues} & E ::= \alpha \,\big|\, [E, E] \,\big|\, \mathit{fst}(E) \,\big|\, \mathit{snd}(E) \,\big|\, v^\bullet \\ \text{Contexts} & e ::= E^\diamond \,\big|\, \tilde{\mu}x.c \\ \text{Expressions} & v ::= x \,\big|\, \mu\alpha.c \,\big|\, \mu x^\bullet.c \,\big|\, \ldots \end{array}$$

Note that focalisation is now on the left, giving rise to a syntactic category of *covalues* (that were called applicative contexts in [CH00]).[11]

The rules are all obtained from LKQ by duality:

$$\frac{}{\Gamma\,;\, \alpha : N \vdash \Delta\,,\, \alpha : N} \qquad \frac{\Gamma\,;\, E_1 : N_1 \vdash \Delta}{\Gamma\,;\, \mathit{fst}(E_1) : N_1 \,\&\, N_2 \vdash \Delta}$$

$$\frac{}{\Gamma\,,\, x : N \vdash x : N \,|\, \Delta} \qquad \frac{\Gamma \vdash v : N \,|\, \Delta \qquad \Gamma \,|\, e : N \vdash \Delta}{\langle v \,|\, e\rangle : (\Gamma \vdash \Delta)} \qquad \ldots$$

We would have arrived to this logic naturally if we had chosen in Section 3 to present LK with a reversible disjunction on the right and an irreversible conjunction on the left, and in Section 4 to present a focalisation discipline with focusing on formulas on the left.

In LKT we can define the following derived CBN implication and terms:

$$M \to^n N = (\neg^- M) \parr N$$
$$\lambda x.v = \mu(x^\bullet, \alpha).\langle v \,|\, \alpha^\diamond\rangle \qquad v_1 v_2 = \mu\alpha.\langle v_1 \,|\, (v_2^\bullet, \alpha)^\diamond\rangle$$

The translation extends to $\lambda\mu$-calculus [Par92] and factors though the $\overline{\lambda}\mu\tilde{\mu}_T$-calculus of [CH00], defining $v \cdot E$ as $(v^\bullet, E)$. Note that the covalues involved in executing call-by-name λ-calculus are just *stacks* of expressions (cf. Section 2).

[10] In [CH00] we also had a difference operator $B - A$ (dual to implication), and two associated introduction operations, whose encodings in the present syntax are $\beta\lambda.e = \tilde{\mu}(\beta^\bullet, x).\langle x^\diamond \,|\, e\rangle$ and $e \cdot V = (e^\bullet, V)$.

[11] Note also that the duality sends a command $\langle v \,|\, e\rangle$ to a command $\langle e' \,|\, v'\rangle$ where v', e' are the mirror images of v, e.

Fig. 3. Translation of LKQ into the λ-calculus / NJ

Translation of formulas:

$$X_{cps} = X \qquad (\neg^+ P)_{cps} = R^{P_{cps}}$$
$$(P \otimes Q)_{cps} = (P_{cps}) \times (Q_{cps}) \quad P \oplus Q_{cps} = (P_{cps}) + (Q_{cps})$$

Translation of terms:

$$\langle v \,|\, e \rangle_{cps} = (v_{cps})(e_{cps}) \qquad (V^\Diamond)_{cps} = \lambda k.k(V_{cps}) \qquad (\mu\alpha.c)_{cps} = \lambda k_\alpha.(c_{cps}) = (\tilde{\mu}\alpha^\bullet.c)_{cps}$$
$$x_{cps} = x \qquad (V_1, V_2)_{cps} = ((V_1)_{cps}, (V_2)_{cps}) \qquad inl(V_1)_{cps} = inl((V_1)_{cps}) \qquad inr(V_2)_{cps} = inr((V_2)_{cps}) \qquad (e^\bullet)_{cps} = e_{cps}$$
$$\alpha_{cps} = k_\alpha \qquad (\tilde{\mu}x.c)_{cps} = \lambda x.(c_{cps}) \qquad (\tilde{\mu}(x_1, x_2).c)_{cps} = \lambda(x_1, x_2).(c_{cps})$$
$$(\tilde{\mu}[inl(x_1).c_1 | inr(x_2).c_2])_{cps} = \lambda z.\mathit{case}\ z\ [inl(x_1) \mapsto (c_1)_{cps}, inr(x_2) \mapsto (c_2)_{cps}]$$

With these definitions, we have:

$$\langle \lambda x.v_1 \,|\, (v_2 \cdot E)^\Diamond \rangle = \left\langle \mu(x^\bullet, \alpha).\langle v_1 \,|\, \alpha^\Diamond \rangle \;\middle|\; (v_2^\bullet, E)^\Diamond \right\rangle \longrightarrow \langle v_1[v_2/x] \,|\, E^\Diamond \rangle$$
$$\langle v_1 v_2 \,|\, E^\Diamond \rangle = \left\langle \mu\alpha.\langle v_1 \,|\, (v_2^\bullet, \alpha)^\Diamond \rangle \;\middle|\; E^\Diamond \right\rangle \longrightarrow \langle v_1 \,|\, (v_2^\bullet, E)^\Diamond \rangle = \langle v_1 \,|\, (v_2 \cdot E)^\Diamond \rangle$$

We are thus back on our feet (cf. section 1)!

Translating LKQ into NJ. Figure 3 presents a translation from LKQ to intuitionistic natural deduction NJ, or, via Curry-Howard, to λ-calculus extended with products and sums. In the translation, R is a fixed target formula (cf. Section 2). We translate $(\neg^+ \_)$ as "$\_$ implies R" (cf. [Kri91, LRS93]). We write B^A for function types / intuitionistic implications. The rules of L_{foc} are simulated by β-reductions. One may think of the source L_{foc} terms as a description of the target ones "in direct style" (cf. [Dan94]).

Proposition 3. *We set* $\Gamma_{cps} = \{x : P_{cps} \mid x : P \in \Gamma\}$ $\qquad R^{\Delta_{cps}} = \{k_\alpha : R^{P_{cps}} \mid \alpha : P \in \Delta\}$. *We have:*

$c : (\Gamma \vdash \Delta)$	$\Gamma \vdash V : P\,;\, \Delta$	$\Gamma \vdash v : P \,|\, \Delta$	$\Gamma \,|\, e : P \vdash \Delta$
$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$
$\Gamma_{cps}\ ,\ R^{\Delta_{cps}} \vdash c_{cps} : R$	$\Gamma_{cps}\ ,\ R^{\Delta_{cps}} \vdash V_{cps} : P_{cps}$	$\Gamma_{cps}\ ,\ R^{\Delta_{cps}} \vdash v_{cps} : R^{R^{P_{cps}}}$	$\Gamma_{cps}\ ,\ R^{\Delta_{cps}} \vdash e_{cps} : R^{P_{cps}}$

Moreover, the translation preserves reduction: if $t \longrightarrow t'$, *then* $t_{cps} \longrightarrow^* (t')_{cps}$.

Composing the previous translations, from CBN $\lambda\mu$-calculus to LKT then through duality to LKQ then to NJ, what we obtain is the CPS translation due to Lafont, Reus, and Streicher [LRS93] (LRS translation, for short).

Translating LKQ into LLP. The translation just given from LKQ to NJ does actually two transformations for the price of one: *from classical to intuitionistic*, and *from sequent calculus style to natural deduction style*. The intermediate target and source of this decomposition is nothing but a subsystem of Laurent's polarised linear logic LLP [Lau02] [12]. We adopt a presentation of LLP in which all negative formulas are handled as positive formulas on the left, and hence in which $!N$ and $?P$ are replaced by $\neg^+$ with the appropriate change of side. With these conventions, LLP is nothing but the system called LJ_0 in [Lau09]. For the purpose of giving a system L term syntax, we distinguish three kinds of sequents for our subsystem of LLP:

$$(\Gamma \vdash) \qquad (\Gamma \vdash P\,;\,) \qquad (\Gamma \,|\, P \vdash)$$

The syntax, the computation rules, and the typing rules are as follows (omitting explicit substitutions):

$$c ::= \langle V \,|\, e \rangle \quad V ::= x \;|\; e^\bullet \;|\; (V, V) \;|\; inl(V) \;|\; inr(V)$$
$$e ::= V^\Diamond \;|\; \tilde{\mu}x.c \;|\; \tilde{\mu}(x_1, x_2).c \;|\; \tilde{\mu}[inl(x_1).c_1 | inr(c_2).c_2]$$

$$\langle V \,|\, \tilde{\mu}x.c \rangle \longrightarrow c[V/x]$$
$$\langle e^\bullet \,|\, V^\Diamond \rangle \longrightarrow \langle V \,|\, e \rangle$$
$$\langle (V_1, V_2) \,|\, \tilde{\mu}(x_1, x_2).c \rangle \longrightarrow c[V_1/x_1, V_2/x_2]$$
$$\langle inl(V_1) \,|\, \tilde{\mu}[inl(x_1).c_1 | inr(x_2).c_2] \rangle \longrightarrow c_1[V_1/x_1] \qquad \langle inr(V_2) \,|\, \tilde{\mu}[inl(x_1).c_1 | inr(x_2).c_2] \rangle \longrightarrow c_2[V_2/x_2]$$

[12] Specifically, no positive formula is allowed on the right in the rules for $⅋$ and $\&$ and in the right premise of the cut rule.

$$\frac{}{\Gamma\,,\ x:P\vdash x:P\,;}\qquad \frac{\Gamma\vdash V:P\,;\qquad \Gamma\,|\,e:P\vdash}{\langle V\,|\,e\rangle:(\Gamma\vdash)}\qquad \frac{c:(\Gamma\,,\ x:P\vdash)}{\Gamma\,|\,\tilde{\mu}x.c:P\vdash}$$

$$\frac{\Gamma\,|\,e:P\vdash}{\Gamma\vdash e^{\bullet}:\neg^{+}P\,;}\qquad \frac{\Gamma\vdash V_1:P_1\,;\qquad \Gamma\vdash V_2:P_2\,;}{\Gamma\vdash(V_1,V_2):P_1\otimes P_2\,;}\qquad \frac{\Gamma\vdash V_1:P_1\,;}{\Gamma\vdash inl(V_1):P_1\oplus P_2\,;}\qquad \frac{\Gamma\vdash V_2:P_2\,;}{\Gamma\vdash inr(V_2):P_1\oplus P_2\,;}$$

$$\frac{\Gamma\vdash V:P\,;}{\Gamma\,|\,V^{\diamond}:\neg^{+}P\vdash}\qquad \frac{c:(\Gamma,x_1:P_1,x_2:P_2\vdash)}{\Gamma\,|\,\tilde{\mu}(x_1,x_2).c:P_1\otimes P_2\vdash}\qquad \frac{c_1:(\Gamma,x_1:P_1\vdash)\qquad c_2:(\Gamma,x_2:P_2\vdash)}{\Gamma\,|\,\tilde{\mu}[inl(x_1).c_1|inr(x_2).c_2]:P_1\oplus P_2\vdash}$$

The constructs $e^{\bullet}$ and $V^{\diamond}$ transcribe LLP's *promotion* and *dereliction* rule, respectively. The compilation from LKQ to (the subsystem of) LLP turns every $\alpha:P$ on the right to a $k_\alpha:\neg^{+}P$ on the left. We write $\neg^{+}(\ldots,\alpha:P,\ldots)=(\ldots,k_\alpha:\neg^{+}P,\ldots)$. The translation is as follows (we give only the non straightforward cases):

$$\begin{array}{l}\langle v\,|\,e\rangle_{\mathsf{LLP}}=\langle(e_{\mathsf{LLP}})^{\bullet}\,|\,v_{\mathsf{LLP}}\rangle\\ (\mu\alpha.c)_{\mathsf{LLP}}=\tilde{\mu}k_\alpha.c_{\mathsf{LLP}}\qquad (V^{\diamond})_{\mathsf{LLP}}=(V_{\mathsf{LLP}})^{\diamond}\\ \alpha_{\mathsf{LLP}}=\tilde{\mu}x.\langle k_\alpha\,|\,x^{\diamond}\rangle\quad (\tilde{\mu}\alpha^{\bullet}.c)_{\mathsf{LLP}}=\tilde{\mu}k_\alpha.(c_{\mathsf{LLP}})\end{array}$$

Note that we can optimise the translation of $\langle V^{\diamond}\,|\,e\rangle$, and (up to an expansion rule) of $V^{\blacklozenge}=\tilde{\mu}\alpha^{\bullet}.\langle V^{\diamond}\,|\,\alpha\rangle$:

$$\begin{array}{l}\langle V^{\diamond}\,|\,e\rangle_{\mathsf{LLP}}=\langle(e_{\mathsf{LLP}})^{\bullet}\,|\,(V_{\mathsf{LLP}})^{\diamond}\rangle\longrightarrow\langle V_{\mathsf{LLP}}\,|\,e_{\mathsf{LLP}}\rangle\\ (V^{\blacklozenge})_{\mathsf{LLP}}=\tilde{\mu}k_\alpha.\langle V_{\mathsf{LLP}}\,|\,\alpha_{\mathsf{LLP}}\rangle\ =\tilde{\mu}k_\alpha.\big\langle V_{\mathsf{LLP}}\,\big|\,\tilde{\mu}x.\langle k_\alpha\,|\,x^{\diamond}\rangle\big\rangle\longrightarrow\tilde{\mu}k_\alpha.\langle k_\alpha\,|\,(V_{\mathsf{LLP}})^{\diamond}\rangle=(V_{\mathsf{LLP}})^{\diamond}\end{array}$$

These optimisations allow us to define a right inverse to $_{\mathsf{LLP}}$ (that maps $V^{\diamond}$ to $V^{\blacklozenge}$)), i.e.:

LLP *(restricted as above) appears as a retract of* LKQ.

The translation simulates reductions and is well typed:

$$\begin{array}{lcl}c:(\Gamma\vdash\Delta) & \Rightarrow & c_{\mathsf{LLP}}:(\Gamma,\neg^{+}\Delta\vdash)\\ \Gamma\vdash v:P\,|\,\Delta & \Rightarrow & \Gamma,\neg^{+}\Delta\,|\,v_{\mathsf{LLP}}:\neg^{+}P\vdash\\ \Gamma\vdash V:P\,;\,\Delta & \Rightarrow & \Gamma,\neg^{+}\Delta\vdash V_{\mathsf{LLP}}:P\,;\\ \Gamma\,|\,e:P\vdash\Delta & \Rightarrow & \Gamma,\neg^{+}\Delta\,|\,e_{\mathsf{LLP}}:P\vdash\end{array}$$

We note that this compilation blurs the distinction between a continuation variable and an ordinary variable (like in the classical CPS translations).

Example 4. The classical proof $(\,|\,\tilde{\mu}\beta^{\bullet}.\langle(\alpha^{\bullet})^{\diamond}\,|\,\beta\rangle:\neg^{+}\neg^{+}P\vdash\alpha:P)$ is translated (using the above optimisation) to the intuitionistic proof $(k_\alpha:\neg^{+}P\,|\,((\tilde{\mu}x.\langle k_\alpha\,|\,x^{\diamond}\rangle)^{\bullet})^{\diamond}:\neg^{+}\neg^{+}P\vdash)$. Without term decorations, we have turned a proof of the classically-only provable sequent $(\,|\,\neg^{+}\neg^{+}P\vdash P)$ into an intuitionistic proof of $(\neg^{+}P\,|\,\neg^{+}\neg^{+}P\vdash)$.

All what is left to do in order to reach then NJ[13] from (our subsystem of) LLP is to turn contexts $e:P$ into values of type $\neg^{+}P$, and to rename $\neg^{+}\_$, $\otimes$, and $\oplus$ as R-, $\times$, and $+$, respectively. More precisely, we describe the target syntax (a subset of a λ-calculus with sums and products) as follows:

$$c::=VV\qquad V::=x\,\big|\,(V,V)\,\big|\,inl(V)\,\big|\,inr(V)\,\big|\,\lambda x.c\,\big|\,\lambda(x_1,x_2).c\,\big|\,\lambda z.\mathit{case}\ z\ [inl(x_1)\mapsto c_1,inr(x_2)\mapsto c_2]$$

Again, we give only the non trivial cases of the translation:

$$\begin{array}{l}\langle V\,|\,e\rangle_{\mathsf{NJ}}=(e_{\mathsf{NJ}})(V_{\mathsf{NJ}})\qquad (e^{\bullet})_{\mathsf{NJ}}=e_{\mathsf{NJ}}\qquad (V^{\diamond})_{\mathsf{NJ}}=\lambda k.k(V_{\mathsf{NJ}})\\ (\tilde{\mu}x.c)_{\mathsf{NJ}}=\lambda x.(c_{\mathsf{NJ}})\quad (\tilde{\mu}(x_1,x_2).c)_{\mathsf{NJ}}=\lambda(x_1,x_2).(c_{\mathsf{NJ}})\\ (\tilde{\mu}[inl(x_1).c_1|inr(c_2).c_2])_{\mathsf{NJ}}=\lambda z.\mathit{case}\ z\ [inl(x_1)\mapsto((c_1)_{\mathsf{NJ}}),inr(x_2)\mapsto(c_2)_{\mathsf{NJ}}]\end{array}$$

[13] In fact, the target of the translation uses only implications of the form R^{P}. Seeing R as "false", this means that the target is in fact intuitionistic logic with conjunction, disjunction and negation in natural deduction style.

Proposition 4. *For all* L_{foc} *term* t *(where* $t ::= c \,|\, V \,|\, v \,|\, e$*), we have:*

$$t_{cps} =_{\beta\eta} (t_{\mathsf{LLP}})_{\mathsf{NJ}} \ .$$

PROOF. We treat the non trivial cases:

$\langle v \,|\, e\rangle_{\mathsf{LLP,NJ}} = \langle (e_{\mathsf{LLP}})^\bullet \,|\, v_{\mathsf{LLP}}\rangle_{\mathsf{NJ}} = (v_{\mathsf{LLP,NJ}})(((e_{\mathsf{LLP}})^\bullet)_{\mathsf{NJ}}) = (v_{\mathsf{LLP,NJ}})(e_{\mathsf{LLP,NJ}})$
$(V^\diamond)_{\mathsf{LLP,NJ}} = ((V_{\mathsf{LLP}})^\diamond)_{\mathsf{NJ}} = \lambda k.k(V_{\mathsf{LLP,NJ}})$
$\alpha_{\mathsf{LLP,NJ}} = (\tilde{\mu}x.\langle k_\alpha \,|\, x^\diamond\rangle)_{\mathsf{NJ}} = \lambda x.(\langle k_\alpha \,|\, x^\diamond\rangle_{\mathsf{NJ}}) = \lambda x.((x^\diamond)_{\mathsf{NJ}})k_\alpha = \lambda x.(\lambda k.kx)k_\alpha =_\beta \lambda x.k_\alpha x =_\eta k_\alpha$ □

Note that, in the proof of the above proposition, the β step is a typical "administrative reduction", so that morally the statement of the proposition holds with η only.

A short foray into linear logic. We end this section by placing the so-called Girard's translation of the call-by-name λ-calculus to linear logic in perspective. The target of this translation is in fact the polarised fragment LL_{pol} of linear logic, obtained by restriction to the polarised formulas:

$$P ::= X \,\big|\, P \otimes P \,\big|\, P \oplus P \,\big|\, !N \qquad\qquad N ::= X^\perp \,\big|\, N \parr N \,\big|\, N \,\&\, N \,\big|\, ?P$$

This fragment is also a fragment of LLP, (cf. [Lau02]), up to the change of notation for the formulas: we write here $\overline{P}$ for $P^\perp$ and $\neg^+\overline{N}$ for $!N$. Girards translation encodes call-by-name implication as follows:

$$(A \to B)^* = !(A^*) \multimap B^* = (?(A^*)^\perp) \parr B^*$$

and then every λ-term $\Gamma \vdash M : A$ into a proof of $!(\Gamma)^* \vdash A^*$. Up to the inclusion of LL_{pol} into LLP, up to the definition of the λ-calculus inside LKT given above, up to the change of notation, and up to the duality between LKT and LKQ, Girard's translation coincides with the (restriction of) our translation above from LKT to LLP. On the other hand, the restriction to the $\lambda\mu$-calculus of our translation from LKT to NJ is the CPS translation of Lafont-Reus-Streicher. Thus, restricted to the λ-calculus, Proposition 4 reads as follows:

Lafont-Reus-Streicher's CPS factors through Girard's translation.

Explicitly, on types, we have that A^* coincides with A as expanded in LKT, and (cf. [CH00]), starting from the simply-typed λ-term $(\Gamma \vdash M : A)$,

- we view M as an expression $(\Gamma \vdash M : A \,|\,)$ of LKT (using the CBN encoding of implication),
- and then as a context $(\,|\, M : \overline{A} \vdash \overline{\Gamma})$ of LKQ,
- then by our above translation we find back the result of Girard's translation $(\neg^+(\overline{\Gamma}) \,|\, M_{\mathsf{LLP}} : \overline{A} \vdash)$,
- and we arrive finally at the Hofmann-Streicher CPS-transform $(\neg^+(\overline{\Gamma}) \vdash M_{cps} : \neg^+(\overline{A}))$ of M, through the translation $_{\mathsf{NJ}}$.[14]

But the above reading is actually stronger, because it is not hard to describe a translation $_{\mathsf{LJ}}$ inverse to $_{\mathsf{NJ}}$, so that up to this further isomorphism, we have that:

The LRS translation of the CBN λ-calculus coincides with Girard's translation.

This nice story does not extend immediately to the $\lambda\mu$-calculus, for which the simplest extension of Girard's translation, taking $\Gamma \vdash M : A \,|\, \Delta$ to a proof of $!(\Gamma^*) \vdash A^*, ?(\Delta)^*$ is not polarised. In fact, Laurent [Lau02] has shown that the natural target for an extension of Girard's translation to CBN $\lambda\mu$-calculus is LLP, in which we can spare the ?'s on the right, i.e., we can translate $\Gamma \vdash M : A \,|\, \Delta$ into a proof of $!(\Gamma^*) \vdash A^*, \Delta^*$ (contractions on negative formulas are free in LLP). So, the extension of the picture to call-by-name $\lambda\mu$-calculus is[15]:

The LRS translation of the CBN $\lambda\mu$-calculus coincides with Laurent-Girard's translation into LLP.

[14] The LRS translation of implication goes as follows: $(A \to B)_{\text{LRS}} = R^{A_{\text{LRS}}} \times B_{\text{LRS}}$, and we have $(\overline{A})_{cps} = A_{\text{LRS}}$.
[15] See also [LR03] for further discussion.

6 A synthetic system

In this section we pursue two related goals.

1. We want to account for the full (or strong) focalisation (cf. [QT96]), which consists in removing the use of contractions in the negative phases and carrying these phases maximally, up to having only atoms on the left of the sequent. The positive phases are made also "more maximal" by allowing the use of the axiom only on positive atoms X. This is of interest in a proof search perspective, since the stronger discipline further reduces the search space.
2. We would like our syntax to quotient proofs over the order of decomposition of negative formulas. The use of structured pattern-matching (cf. Examples 1, 2) is relevant, as we can describe the construction of a proof of $(\Gamma, x : (P_1 \otimes P_2) \otimes (P_3 \otimes P_4) \vdash \Delta)$ out of a proof of $c : (\Gamma, x_1 : P_1, x_2 : P_2, x_3 : P_3, x_4 : P_4 \vdash \Delta)$ "synthetically", by writing $\langle x^\diamond \mid \tilde{\mu}((x_1, x_2), (x_3, x_4)).c\rangle$, where $\tilde{\mu}((x_1, x_2), (x_3, x_4)).c$ stands for an abbreviation of either of the following two commands:

$$\left\langle x^\diamond \,\middle|\, \tilde{\mu}(y, z).\left\langle y^\diamond \,\middle|\, \tilde{\mu}(x_1, x_2).\langle z^\diamond \mid \tilde{\mu}(x_3, x_4).c\rangle\right\rangle\right\rangle \qquad \left\langle x^\diamond \,\middle|\, \tilde{\mu}(y, z).\left\langle z^\diamond \,\middle|\, \tilde{\mu}(x_3, x_4).\langle y^\diamond \mid \tilde{\mu}(x_1, x_2).c\rangle\right\rangle\right\rangle$$

The two goals are connected, since applying strong focalisation will forbid the formation of these two terms (because y, z are values appearing with non atomic types), keeping the synthetic form only... provided we make it first class.

We shall proceed in *two steps*. The first, intermediate one consists in introducing first-class *counterpatterns* and will serve goal 1 but not quite goal 2:

Simple commands	$c ::= \langle v \mid e\rangle$	Commands	$C ::= c \mid [C\ ^{q,q}\ C]$
Expressions	$v ::= V^\diamond \mid \mu\alpha.C$	Values	$V ::= x \mid (V, V) \mid inl(V) \mid inr(V) \mid e^\bullet$
Contexts	$\boxed{e ::= \alpha \mid \tilde{\mu}q.C}$	Counterpatterns	$\boxed{q ::= x \mid \alpha^\bullet \mid (q, q) \mid [q, q]}$

The counterpatterns are to be thought of as constructs that match patterns (see below).

In this syntax, we have gained a unique $\tilde{\mu}$ binder, but the price to pay (provisionally) is that now commands are trees of copairings $[\_\ ^{q_1,q_2}\ \_]$ whose leaves are simple commands.

The typing discipline is restricted with respect to that of Figure 1 (and adapted to the setting with explicit counterpatterns). Let $\Xi = x_1 : X_1, \ldots, x_n : X_n$ denote a left context consisting of *atomic formulas only*. The rules are as follows:

$$\frac{}{\Xi\ ,\ x : X \vdash x : X\ ;\ \Delta} \qquad \frac{}{\Xi \mid \alpha : P \vdash \alpha : P\ ,\ \Delta} \qquad \frac{\Xi \vdash v : P \mid \Delta \qquad \Xi \mid e : P \vdash \Delta}{\langle v \mid e\rangle : (\Xi \vdash \Delta)}$$

$$\frac{C : (\Xi\ ,\ q : P \vdash \Delta)}{\Xi \mid \tilde{\mu}q.C : P \vdash \Delta} \qquad \frac{C : (\Xi \vdash \alpha : P\ ,\ \Delta)}{\Xi \vdash \mu\alpha.C : P \mid \Delta} \qquad \frac{\Xi \vdash V : P\ ;\ \Delta}{\Xi \vdash V^\diamond : P \mid \Delta}$$

$$\frac{\Xi \mid e : P \vdash \Delta}{\Xi \vdash e^\bullet : \neg^+ P\ ;\ \Delta} \qquad \frac{\Xi \vdash V_1 : P_1\ ;\ \Delta \qquad \Xi \vdash V_2 : P_2\ ;\ \Delta}{\Xi \vdash (V_1, V_2) : P_1 \otimes P_2\ ;\ \Delta} \qquad \frac{\Xi \vdash V_1 : P_1\ ;\ \Delta}{\Xi \vdash inl(V_1) : P_1 \oplus P_2\ ;\ \Delta} \qquad \frac{\Xi \vdash V_2 : P_2\ ;\ \Delta}{\Xi \vdash inr(V_2) : P_1 \oplus P_2\ ;\ \Delta}$$

$$\frac{C : (\Gamma \vdash \alpha : P\ ,\ \Delta)}{C : (\Gamma\ ,\ \alpha^\bullet : \neg^+ P \vdash \Delta)} \qquad \frac{C : (\Gamma\ ,\ q_1 : P_1\ ,\ q_2 : P_2 \vdash \Delta)}{C : (\Gamma\ ,\ (q_1, q_2) : P_1 \otimes P_2 \vdash \Delta)} \qquad \frac{C_1 : (\Gamma\ ,\ q_1 : P_1 \vdash \Delta) \qquad C_2 : (\Gamma\ ,\ q_2 : P_2 \vdash \Delta)}{[C_1\ ^{q_1,q_2}\ C_2] : (\Gamma\ ,\ [q_1, q_2] : P_1 \oplus P_2 \vdash \Delta)}$$

Our aim now (*second step*) is to get rid of the tree structure of a command. Indeed, towards our second goal, if $c_{ij} : (\Gamma, x_i : P_i, x_j : P_j \vdash_S \Delta)$ $(i = 1, 2, j = 3, 4)$, we want to identify $[[c_{13}\ ^{x_3,x_4}\ c_{14}]\ ^{x_1,x_2}\ [c_{23}\ ^{x_3,x_4}\ c_{24}]]$ and $[[c_{13}\ ^{x_1,x_2}\ c_{23}]\ ^{x_3,x_4}\ [c_{14}\ ^{x_1,x_2}\ c_{24}]]$. To this effect, we need a last ingredient. We introduce a syntax of *patterns*, and we redefine the syntax of values, as follows:

$$\boxed{\mathcal{V} ::= x \mid e^\bullet \qquad V ::= p\,\langle \mathcal{V}_i / i \mid i \in p\rangle \qquad p ::= x \mid \alpha^\bullet \mid (p, p) \mid inl(p) \mid inr(p)}$$

where $i \in p$ is defined by:

$$\frac{}{x \in x} \quad \frac{}{\alpha^\bullet \in \alpha^\bullet} \quad \frac{i \in p_1}{i \in (p_1, p_2)} \quad \frac{i \in p_2}{i \in (p_1, p_2)} \quad \frac{i \in p_1}{i \in inl(p_1)} \quad \frac{i \in p_2}{i \in inr(p_2)}$$

Moreover, $\mathcal{V}_i$ must be of the form y (resp. $e^\bullet$) if $i = x$ (resp. $i = \alpha^\bullet$).

Patterns are required to be linear, as well as the counterpatterns, for which the definition of "linear" is adjusted in the case $[q_1, q_2]$, in which a variable can occur (but recursively linearly so) in both q_1 and q_2.

Note also that the reformulation of values is up to α-conversion: for example, it is understood that $\alpha^\bullet\langle e^\bullet/\alpha^\bullet\rangle = \beta^\bullet\langle e^\bullet/\beta^\bullet\rangle$

We can now rephrase the logical reduction rules in terms of pattern/counterpattern interaction (whence the terminology), resulting in the following packaging of rules:

$$\frac{V = p\,\langle \ldots y/x, \ldots, e^\bullet/\alpha^\bullet, \ldots\rangle \qquad C[p/q] \longrightarrow^* c}{\langle V^\diamond \mid \tilde{\mu}q.C\rangle \longrightarrow c\{\ldots, y/x, \ldots, e/\alpha, \ldots\}}$$

where $c\{\sigma\}$ is the usual, implicit substitution, and where c (see the next proposition) is the normal form of $C[p/q]$ with respect to the following set of rules:

$$\begin{array}{l} C[(p_1,p_2)/(q_1,q_2),\sigma] \longrightarrow C[p_1/q_1, p_2/q_2, \sigma] \\ [C_1\ {}^{q_1,q_2}\ C_2][inl(p_1)/[q_1,q_2],\sigma] \longrightarrow C_1[p_1/q_1,\sigma] \qquad [C_1\ {}^{q_1,q_2}\ C_2][inr(p_2)/[q_1,q_2],\sigma] \longrightarrow C_2[p_2/q_2,\sigma] \\ C[\alpha^\bullet/\alpha^\bullet,\sigma] \longrightarrow C[\sigma] \qquad C[x/x,\sigma] \longrightarrow C[\sigma] \end{array}$$

Logically, this means that we now consider each formula as made of blocks of *synthetic* connectives.

Example 5. – Patterns for $P = X \otimes (Y \oplus \neg^+Q)$. Focusing on the right yields two possible proof searches:

$$\frac{\Gamma \vdash x'\{\mathcal{V}_{x'}\} : X\,;\,\Delta \quad \Gamma \vdash y'\{\mathcal{V}_{y'}\} : Y\,;\,\Delta}{\Gamma \vdash (x', inl(y'))\{\mathcal{V}_{x'}, \mathcal{V}_{y'}\} : X \otimes (Y \oplus \neg^+Q)\,;\,\Delta} \qquad \frac{\Gamma \vdash x'\{\mathcal{V}_{x'}\} : X\,;\,\Delta \quad \Gamma \vdash \alpha'^\bullet\{\mathcal{V}_{\alpha'\bullet}\} : \neg^+Q\,;\,\Delta}{\Gamma \vdash (x', inr(\alpha'^\bullet))\{\mathcal{V}_{x'}, \mathcal{V}_{\alpha'\bullet}\} : X \otimes (Y \oplus \neg^+Q)\,;\,\Delta}$$

– Counterpattern for $P = X \otimes (Y \oplus \neg^+Q)$. The counterpattern describes the tree structure of P:

$$\frac{c_1 : (\Gamma\ ,\ x : X\ ,\ y : Y \vdash \Delta) \qquad c_2 : (\Gamma\ ,\ x : X\ ,\ \alpha^\bullet : \neg^+Q \vdash \Delta)}{[c_1\ {}^{y,\alpha^\bullet}\ c_2] : (\Gamma\ ,\ (x, [y, \alpha^\bullet]) : X \otimes (Y \oplus \neg^+Q) \vdash \Delta)}$$

We observe that the leaves of the decomposition are in one-to-one correspondence with the patterns p for the (irreversible) decomposition of P on the right:

$$[c_1\ {}^{y,\alpha^\bullet}\ c_2][p_1/q] \longrightarrow^* c_1 \qquad [c_1\ {}^{y,\alpha^\bullet}\ c_2][p_2/q] \longrightarrow^* c_2$$

where $q = (x, [y, \alpha^\bullet])\ ,\ p_1 = (x, inl(y))\ ,\ p_2 = (x, inr(\alpha^\bullet))$.

This correspondence is general. We define two predicates $c \in C$ and $q \perp p$ ("q is orthogonal to p") as follows:

$$\frac{}{c \in c} \qquad \frac{c \in C_1}{c \in [C_1\ {}^{q_1,q_2}\ C_2]} \qquad \frac{c \in C_2}{c \in [C_1\ {}^{q_1,q_2}\ C_2]}$$

$$\frac{}{x \perp x} \qquad \frac{}{\alpha^\bullet \perp \alpha^\bullet} \qquad \frac{q_1 \perp p_1 \quad q_2 \perp p_2}{(q_1, q_2) \perp (p_1, p_2)} \qquad \frac{q_1 \perp p_1}{[q_1, q_2] \perp inl(p_1)} \qquad \frac{q_2 \perp p_2}{[q_1, q_2] \perp inr(p_2)}$$

We can now state the correspondence result.

Proposition 5. *Let $C : (\Xi\ ,\ q : P \vdash \Delta)$ (as in the assumption of the typing rule for $\tilde{\mu}q.C$), and let p be such that q is orthogonal to p. Then the normal form c of $C[p/q]$ is a simple command, and the mapping $p \mapsto c$ (q, C fixed) from $\{p \mid q \perp p\}$ to $\{c \mid c \in C\}$ is one-to-one and onto.*

Fig. 4. The syntax and reduction semantics of $\mathsf{L}_{\text{synth}}$

$$
\begin{array}{lll}
c ::= \langle v \,|\, e \rangle & v ::= V^{\Diamond} \;\big|\; \mu\alpha.c & \\
V ::= p\,\langle \mathcal{V}_i/i \mid i \in p\rangle & \mathcal{V} ::= x \;\big|\; e^{\bullet} & p ::= x \;\big|\; \alpha^{\bullet} \;\big|\; (p,p) \;\big|\; inl(p) \;\big|\; inr(p) \\
e ::= \alpha \;\big|\; \tilde{\mu}q.\{p \mapsto c_p \mid q \perp p\} & & q ::= x \;\big|\; \alpha^{\bullet} \;\big|\; (q,q) \;\big|\; [q,q]
\end{array}
$$

$$
\boxed{(\tilde{\mu}^{+})\;\; \langle (p\,\langle \ldots, y/x, \ldots, e^{\bullet}/\alpha^{\bullet} \ldots\rangle)^{\Diamond} \,|\, \tilde{\mu}q.\{p \mapsto c_p \mid q \perp p\}\rangle \longrightarrow c_p\,\{\ldots, y/x, \ldots, e/\alpha, \ldots\rangle\}} \quad \boxed{(\mu)\;\; \langle \mu\alpha.c \,|\, e\rangle \longrightarrow c\{e/\alpha\}}
$$

Typing rules: the old ones for $\alpha, x, e^{\bullet}, c$, plus the following ones:

$$
\frac{\ldots \qquad \Gamma \vdash \mathcal{V}_i : P_i \,;\, \Delta \quad ((i : P_i) \in \Gamma(p,P)) \qquad \ldots}{\Gamma \vdash p\,\langle \mathcal{V}_i/i \mid i \in p\rangle : P \,;\, \Delta}
\qquad\qquad
\frac{\ldots \qquad c_p : (\Gamma\,,\, \Xi(p,P) \vdash \Delta(p,P)\,,\, \Delta) \quad (q \perp p) \qquad \ldots}{\Gamma \,|\, \tilde{\mu}q.\{p \mapsto c_p \mid q \perp p\} : P \vdash \Delta}
$$

where $\Gamma(p,P)$ must be successfully defined as follows:

$$
\begin{array}{l}
\Gamma(x,X) = (x : X) \qquad \Gamma(\alpha^{\bullet}, \neg^{+}P) = (\alpha^{\bullet} : \neg^{+}P) \\
\Gamma((p_1,p_2), P_1 \otimes P_2) = \Gamma(p_1,P_1)\,,\,\Gamma(p_2,P_2) \qquad \Gamma(inl(p_1), P_1 \oplus P_2) = \Gamma(p_1,P_1) \qquad \Gamma(inr(p_2), P_1 \oplus P_2) = \Gamma(p_2,P_2)
\end{array}
$$

and where

$$
\Xi(p,P) = \{x : X \mid x : X \in \Gamma(p,P)\} \qquad \Delta(p,P) = \{a : P \mid \alpha^{\bullet} : \neg^{+}P \in \Gamma(p,P)\}
$$

PROOF. The typing and the definition of orthogonality entail that in all intermediate $C[\sigma]$'s the substitution σ has an item for each counterpattern in the sequent, and that reduction progresses. The rest is easy. (Note that a more general statement is needed for the induction to go through, replacing $\Xi\,,\, q : P$, "q orthogonal to p", and $C[p/q]$ with $\Xi\,,\, q_1 : P_1\,,\,\ldots\,,\, q_n : P_n$, "$q_i$ orthogonal to p_i for $i = 1, \ldots, n$", and $C[p_1/q_1, \ldots p_n/q_n]$, respectively.) □

Thanks to this correspondence, we can quotient over the "bureaucracy" of commands, and we arrive at the calculus described in Figure 4, together with its typing rules, which we call *synthetic system* L, or $\mathsf{L}_{\text{synth}}$. The $\tilde{\mu}$ construct of $\mathsf{L}_{\text{synth}}$ is closely related to Zeilberger's higher-order abstract approach to focalisation in [Zei08]: indeed we can view $\{p \mapsto c \mid q \perp p\}$ as a function from patterns to commands. We actually prefer to see here a ***finite*** record whoses fields are the p's orthogonal to q. There are only two reduction rules in $\mathsf{L}_{\text{synth}}$. the μ-rule now expressed with implicit substitution and the $\tilde{\mu}^{+}$-rule, which combines two familiar operations: select a field p (like in object-oriented programming), and substitute (like in functional programming). The next proposition relates $\mathsf{L}_{\text{synth}}$ to L_{foc}.

Proposition 6. *The typing system of* $\mathsf{L}_{\text{synth}}$ *is complete*[16] *with respect to* LKQ.

PROOF. The completeness of $\mathsf{L}_{\text{synth}}$ with respect to the intermediate system above is an easy consequence of Proposition 5. We are thus left with proving the completeness of the intermediate system. We define a rewriting relation between sets of sequents as follows:

$$
\begin{array}{l}
(\Gamma, x : \neg^{+}P \vdash \Delta), \mathbf{S} \rightsquigarrow (\Gamma \vdash \alpha : P, \Delta), \mathbf{S} \\
(\Gamma, x : P_1 \otimes P_2 \vdash \Delta), \mathbf{S} \rightsquigarrow (\Gamma, x_1 : P_1, x_2 : P_2 \vdash \Delta), \mathbf{S} \\
(\Gamma, x : P_1 \oplus P_2 \vdash \Delta), \mathbf{S} \rightsquigarrow (\Gamma, x_1 : P_1 \vdash \Delta), (\Gamma, x_2 : P_2 \vdash \Delta), \mathbf{S}
\end{array}
$$

(where α, x_1, x_2 are fresh). A normal form for this notion of reduction is clearly a set of sequents of the form $\Xi \vdash \Delta$. It is also easy to see that $\rightsquigarrow$ is confluent and (strongly) normalising. In what follows, $\vdash_S$ (resp. $\vdash$) will signal the intermediate proof system (resp. L_{foc}). The following property is easy to check.

If $(\Xi_1 \vdash \Delta_1), \ldots, (\Xi_n \vdash \Delta_n)$ is the normal form of $(x_1 : P_1, \ldots, x_m : P_m \vdash \Delta)$ for $\rightsquigarrow$ and if $c_i : (\Xi_i \vdash_S \Delta_i)$, then there exist $q_1, \ldots, q_m$ and a command $C : (q_1 : P_1, \ldots, q_m : P_m \vdash_S \Delta)$ whose leaves are the c_i's.

We prove the following properties together:

1) If $c : (x_1 : P_1, \ldots, x_m : P_m \vdash \Delta)$, then there exist $q_1, \ldots, q_m$ and C such that $C : (q_1 : P_1, \ldots, q_m : P_m \vdash_S \Delta)$.
2) If $\Xi \,|\, e : P \vdash \Delta$, then there exists e' such that $\Xi \,|\, e' : P \vdash_S \Delta$ (and similarly for expressions v).

[16] It is also easy to see that the underlying translation from L_{foc} to $\mathsf{L}_{\text{synth}}$ is reduction-reflecting (cf. Remark 3).

The proof is by induction on a notion of size which is the usual one except that the size of a variable x is not 1 but the size of its type. It is easy to check that the substitutions involved in Lemma 1 do not increase the size. The interesting case is $c = \langle v \,|\, e \rangle$. Let $(\Xi_1 \vdash \Delta_1), \ldots, (\Xi_n \vdash \Delta_n)$ be the normal form of $(x_1 : P_1, \ldots, x_m : P_m \vdash \Delta)$. Then, by repeated application of Lemma 1, we get $v_1, \ldots, v_n$ and $e_1, \ldots, e_n$, and then by induction $v'_1, \ldots, v'_n$ and $e'_1, \ldots, e'_n$ which we assemble pairwise to form $\langle v'_1 \,|\, e'_1 \rangle, \ldots, \langle v'_n \,|\, e'_n \rangle$, which in turn, as noted above, can be assembled in a tree $C : (q_1 : P_1, \ldots, q_m : P_m \vdash_S \Delta)$. □

Putting together Propositions 1 and 6, we have proved that $\mathsf{L}_{\text{synth}}$ is complete with respect to LK for provability.

Remark 5. – In the multiplicative case (no C, $inl(V)$, $inr(V)$, $[q_1, q_2]$), there is a unique p such that $q \perp p$, namely q, and the syntax boils down to

$$\mathcal{V} ::= x \,\big|\, e^\bullet \quad V ::= p\,\langle \mathcal{V}_i / i \mid i \in p \rangle \quad v ::= x \,\big|\, \tilde{\mu} q.\{c\} \quad c ::= \langle V^\diamond \,|\, \alpha \rangle$$

Compare with *Böhm trees*: $M ::= \overbrace{\lambda \boldsymbol{x}.\, \underbrace{P}_{c}}^{e}$ $\qquad$ $P ::= y\, \underbrace{\overbrace{M_1}^{\mathcal{V}} \ldots \overbrace{M_n}^{\mathcal{V}}}_{V}$. For example (cf. the CBN translation in Section 5), if M_j translates to e_j, then $\lambda x_1 x_2 . x M_1 M_2 M_3$ translates to $\tilde{\mu}(\tilde{x_1}^\bullet, \tilde{x_2}^\bullet, y).\langle p \langle \mathcal{V}_i / i \mid i \in p \rangle^\diamond \,|\, \tilde{x} \rangle$, where $p = (\alpha_1^\bullet, \alpha_2^\bullet, \alpha_3^\bullet, z)$, $\mathcal{V}_{z_j} = (e_j)^\bullet$, and $\mathcal{V}_z = y$.

– (for readers familiar with [Gir01]) Compare with the syntax for ludics presented in [Cur06]:

$$\overbrace{M}^{e} ::= \{ \overbrace{J}^{p} \mapsto \lambda\{x_j \mid j \in J\}.P_J \mid \overbrace{J \in \mathcal{N}}^{q \perp p} \} \qquad \underbrace{P}_{c} ::= (x \cdot \underbrace{\overbrace{I}^{p}\{M_i \mid i \in I\}}_{V}) \,\big|\, \Omega \,\big|\, \maltese$$

7 Conclusion

We believe that Curien-Herbelin's syntactic kit, which we could call *system* L for short, provides us with a robust infrastructure for proof-theoretical investigations, and for applications in formal studies in operational semantics. Thus, the work presented here is faithful to the spirit of Herbelin's Habilitation Thesis [Her05], where he advocated an incremental approach to connectives, starting from a pure control kernel.

On the proof-theoretical side, we note, with respect to the original setting of ludics [Gir01], that a pattern p is more precise than a ramification I (finite tree of subaddresses vs a set of immediate subaddresses). We might use this additional precision to design a version of ludics where axioms are first-class rather than treated as infinite expansions.

On the side of applications to programming language semantics, the good fit between abstract machines and our syntax L_{foc} makes it a good candidate for being used as an intermediate language appropriate to reason about the correctness of abstract machines (see also [Lev04]). In this spirit, in order to account for languages with mixed call-by-value / call-by-name features, one may give a truly bilateral presentation of L_{foc} that freely mixes positive and negative formulas like in Girard's LC [Gir91].[17] Such a system is presented in the long version of [Mun09].

Finally, we wish to thank Bob Harper, Hugo Herbelin, and Olivier Laurent for helpful discussions.

References

[ACCL92] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy, Explicit Substitutions, J. of Functional Programming 1(4), 375-416 (1992).

[And92] J.-M. Andreoli, Logic programming with focusing proofs in linear logic, J. of Logic and Computation 2(3), 297-347 (1992).

[CH00] P.-L. Curien and H. Herbelin, The duality of computation, Proc. Int. Conf. on Functional Programming 2000, ACM Press.

[Cur06] P.-L. Curien, Introduction to linear logic and ludics, part II, Advances in Mathematics (China) 35 (1), 1-44 (2006).

[DJS97] V. Danos, J.-B. Joinet, H. Schellinx, A new deconstructive logic: linear logic, J. of Symbolic Logic 62(3) 755-807 (1997).

[Dan94] O. Danvy, Back to direct style, Science of Computer Programming, 22(3),183-195 (1994).

[GLF89] J.-Y. Girard, Y. Lafont, and P. Taylor, Proofs and Types, Cambridge University Press (1989).

[17] See also [Mur92] for an early analysis of the computational meaning of LC from a programming language perspective.

[Gir91] J.-Y. Girard, A new constructive logic: classical logic, Math. Struct. in Computer Science 1, 255-296 (1991).
[Gir01] J.-Y. Girard, Locus solum: from the rules of logic to the logic of rules, MSCS 11(3), 301-506 (2001).
[Gri90] T. Griffin, A formulae-as-types notion of control, Proc. ACM Principles of Prog. Lang. (1990).
[Her95] H. Herbelin, Séquents qu'on calcule, Thèse de Doctorat, Université Paris 7, available from `http://pauillac.inria.fr/~herbelin` (1995).
[Her05] H. Herbelin, C'est maintenant qu'on calcule, au cœur de la dualité, Mémoire d'habilitation, available from cited url (2005).
[Kri91] J.-L. Krivine, Lambda-calcul, types et modèles, Masson (1991).
[Kri07] J.-L. Krivine, A call-by-name lambda-calculus machine, Higher Order and Symbolic Computation 20, 199-207 (2007).
[LRS93] Y. Lafont, B. Reus, and T. Streicher, Continuation Semantics or Expressing Implication by Negation, Technical Report, available from `http://iml.univ-mrs.fr/~lafont` (1993).
[Lan64] P. Landin, The mechanical evaluation of expressions, Computer Journal 6, 308-320 (1964).
[Lau04] O. Laurent, A proof of the focalisation property of linear logic, manuscript available from `http://perso.ens-lyon.fr/olivier.laurent` (2004).
[Lau02] O. Laurent, Etude de la polarisation en logique, Thèse de Doctorat, Univ. Aix-Marseille II (2002).
[Lau09] O. Laurent, Intuitionistic dual-intuitionistic nets, submitted, available from cited url.
[LQT05] O. Laurent, M. Quatrini, and L. Tortora de Falco, Polarised and focalised linear and classical proofs, Annals of Pure and Applied Logic, 134, no 2-3,217–264 (2005).
[LR03] O. Laurent and L. Regnier, About translations of classical logic into polarised linear logic, Proc. LICS 2003, ACM Press.
[Lev04] P.B. Levy, Call-by-push-value. A functional/imperative synthesis, Semantic Structures in Computation, Springer (2004).
[Mun09] G. Munch-Maccagnoni, Focalisation and classical realisability, Proc. CSL 2009 LNCS 5771, 409–423, Springer (long version available from `http://perso.ens-lyon.fr/guillaume.munch/articles`).
[Mur92] C. Murthy, A computational analysis of Girard's translation and LC, Proc. LICS 1992.
[Nip93] T. Nipkow, Orthogonal Higher-Order Rewrite Systems are Confluent.Proc. TLCA, LNCS 664 , 306-317 (1993).
[OS97] L. Ong and C. Stewart, A Curry-Howard foundation for functional computation with control, Proc. POPL 97.
[Par92] M. Parigot, $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction, in Proc. of the Int. Conf. on Logic Programming and Automated Reasoning, St. Petersburg, LNCS 624 (1992).
[Plo75] G. Plotkin, Call-by-name, call-by-value and the lambda-calculus, Theoretical Computer Science 1, 125-159 (1975).
[QT96] M. Quatrini, L. Tortora de Falco, Polarisation des preuves classiques et renversement, C.R.A.S..323(I),113-116 (1996).
[Roc05] J. Rocheteau, $\lambda\mu$-calculus and duality: call-by-name and call-by-value, Proc. RTA 2005, LNCS 3467.
[Wad03] Ph. Wadler, Call-by-value is dual to call-by-name, Proc. Int. Conf. on Functional Programming (2003).
[Zei08] N. Zeilberger, On the unity of duality, Annals of Pure and Applied Logic 153(1), 66-96 (2008).