The Duality of Computation under Focus

Pierre-Louis Curien[1] and Guillaume Munch-Maccagnoni[2]

[1] CNRS, Paris 7, and INRIA
[2] Paris 7, and INRIA

Abstract. We review the 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 of the duality of computation. We provide a term language for a presentation of LK (with conjunction, disjunction, and negation), and we transcribe cut elimination as (non confluent) rewriting. A key slogan, 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 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 [3].

Section 2 discusses the relation between *abstract machines* for the λ-calculus (extended with control) and (classical) *sequent calculus*. Section 3 presents a language (with a one-to-one correspondence between well-typed terms and proof trees) for a version of LK in which we choose to give a dissymetric presentation for the conjunction on the left and for the disjunction on the right, anticipating a key ingredient of focalisation. 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}) [19] 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.

Finally, in Section 6, further reinforcing of the focalisation discipline 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. 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 (as in [1]) added to the language together with rules propagating it.

C.S. Calude and V. Sassone (Eds.): TCS 2010, IFIP AICT 323, pp. 165–181, 2010.

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 Landin's pioneering [15]).

Here is Krivine machine, a simple device for call-by-name λ-calculus [13]:

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

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:

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

The first rule explains how the continuation E gets *captured*, and the second rule how it gets *restored*. Griffin [9] 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 [23]): $\mathcal{C}(M) = \mu\beta.\langle M \mid \beta^* \cdot [\,] \rangle$, 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: $(\Gamma \vdash M : A \mid \Delta)$, $(\Gamma \mid E : A \vdash \Delta)$, $c : (\Gamma \vdash \Delta)$. 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: $\langle \mu\alpha.c \mid E \rangle \to c\{E/\alpha\}$.

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 [10]:

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

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:

$$\boxed{\langle MN \mid e \rangle \to \langle N \mid M \odot e \rangle \qquad \langle V \mid M \odot e \rangle \to \langle M \mid V \cdot e \rangle}$$

(the operational rule for $\lambda x.M$ is unchanged)[1]. Here, V is a *value*, defined as being either a variable or an abstraction (this goes back to [24]). Again, we can read $M \odot e$ as "a map $V \mapsto \langle M \mid V \cdot e \rangle$", or, introducing a new binder $\tilde{\mu}$ (binding now ordinary variables): $M \odot e = \tilde{\mu}x.\langle M \mid x \cdot e \rangle$. The typing rule for this operator is (left activation): $\dfrac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \mid \tilde{\mu}x.c : A \vdash \Delta}$, and the operational rule is $\langle V \mid \tilde{\mu}x.c \rangle \to c\{V/x\}$ (V value).

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

We have arrived at Curien and Herbelin's $\overline{\lambda}\mu\tilde{\mu}_Q$-calculus [3]:

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

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:

$$\boxed{\begin{array}{l}\langle (\lambda x.M)^\diamond \mid V \cdot e \rangle \to \langle M\{V/x\} \mid e \rangle \\ \langle \mu\alpha.c \mid e \rangle \to c\{e/\alpha\} \qquad \langle V^\diamond \mid \tilde{\mu}x.c \rangle \to c\{V/x\}\end{array}}$$

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

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)

$$\frac{c : (\Gamma \vdash \alpha : A, \Delta)}{\Gamma \,|\, \tilde{\mu}\alpha^{\bullet}.c : \neg A \vdash \Delta} \qquad \frac{c : (\Gamma, x_1 : A_1, x_2 : A_2 \vdash \Delta)}{\Gamma \,|\, \tilde{\mu}(x_1, x_2).c : A_1 \wedge A_2 \vdash \Delta}$$

$$\frac{c_1 : (\Gamma, x_1 : A_1 \vdash \Delta) \qquad c_2 : (\Gamma, x_2 : A_2 \vdash \Delta)}{\Gamma \,|\, \tilde{\mu}[inl(x_1).c_1 | inr(x_2).c_2] : A_1 \vee A_2 \vdash \Delta}$$

(deactivation)

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

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[2]:

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

[2] 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.

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.*[3]

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 $\langle \mu\alpha.\langle v \mid \alpha\rangle \mid \tilde{\mu}x.\langle x \mid e\rangle\rangle$):

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

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

$\langle \mu\alpha.\langle(\tilde{\mu}y.c)^\bullet \mid \beta\rangle \mid \tilde{\mu}x.d\rangle \to \left\langle \mu\beta'.\langle(\tilde{\mu}y.\langle \mu\alpha.c \mid \tilde{\mu}x.d\rangle)^\bullet \mid \beta'\right\rangle \Big| \tilde{\mu}y.\langle y \mid \beta\rangle \Big\rangle$ ($\neg$ right)
(similar rules with the other right introduction rules and with the left introduction rules)

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

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

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

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

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

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

[3] For the same reason, we have three connectives instead of just, 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 *provavility*.

(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):

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

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

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

$$\frac{\Gamma\vdash v_1:A_1\,|\,\Delta}{\Gamma\vdash inl(v_1):A_1\vee A_2\,|\,\Delta}\qquad\frac{\Gamma\vdash v_2:A_2\,|\,\Delta}{\Gamma\vdash inr(v_2):A_1\vee A_2\,|\,\Delta}$$

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

Note that we 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 $\langle v\,|\,e\rangle$ 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)	$\langle\mu\alpha.c\,|\,e\rangle\to c[e/\alpha]\qquad\langle v\,|\,\tilde{\mu}x.c\rangle\to c[v/x]$
(logical)	$\langle e^\bullet\,|\,\tilde{\mu}\alpha^\bullet.c\rangle\to c[e/\alpha]\quad\langle(v_1,v_2)\,|\,\tilde{\mu}(x_1,x_2).c\rangle\to c[v_1/x_1,v_2/x_2]$
	$\langle inl(v_1)\,|\,\tilde{\mu}[inl(x_1).c_1|inr(x_2).c_2]\rangle\to c_1[v_1/x_1]\quad(\text{idem }inr)$
(commutation)	$\langle v\,|\,e\rangle[\sigma]\to\langle v[\sigma]\,|\,e[\sigma]\rangle$
	$x[\sigma]\to x\ \ (x\text{ not declared in }\sigma)\qquad x[v/x,\sigma]\to v\quad(\text{idem }\alpha[\sigma])$
	$(\mu\alpha.c)[\sigma]\to\mu\alpha.(c[\sigma])\quad(\text{idem }(\tilde{\mu}x.c)[\sigma])\quad(\text{capture avoiding})$
	(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[4].

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

[4] The precise relation with the previous rules is as follows: for all s_1, s_2 such that $s_1\to s_2$ in the first system, there exists s such that $s_1\to^* s$ and $s_2\to^* s$ in the new system.

The problem with classical logic viewed as a computational system is its wild non confluence, as captured by Lafont's critical pair [6,4], 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): $c_1 \ {}^*\!\leftarrow \ \langle \mu\alpha.c_1 \,|\, \tilde{\mu}x.c_2 \rangle \ \rightarrow^* \ c_2$. So, all proofs are identified... *Focalisation*, discussed in the next section, will guide us to solve this dilemma.

4 A Syntax for Focalised Classical Logic

We adapt the *focalisation discipline* (originally introduced by [2] in the setting of linear logic) to LK. A focalised proof search alternates between right and left phases:

- *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 [7]). 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 $\vdash \overline{\Gamma}, \Delta$).

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

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

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

$$\frac{c : (\Gamma \vdash \alpha : P, \Delta)}{\Gamma \mid \tilde{\mu}\alpha^{\bullet}.c : \neg^{+}P \vdash \Delta} \qquad \frac{c : (\Gamma, x_1 : P_1, x_2 : P_2 \vdash \Delta)}{\Gamma \mid \tilde{\mu}(x_1, x_2).c : P_1 \otimes P_2 \vdash \Delta}$$

$$\frac{c_1 : (\Gamma, x_1 : P_1 \vdash \Delta) \qquad c_2 : (\Gamma, x_2 : P_2 \vdash \Delta)}{\Gamma \mid \tilde{\mu}[inl(x_1).c_1 | inr(x_2).c_2] : P_1 \oplus P_2 \vdash \Delta}$$

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

Fig. 1. System LKQ

We are now ready to give the syntax of our calculus, which is a variant of the one given by the second author in [19][5].

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

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 [4]). 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 $\tilde{\mu}(x, y).\langle y^{\diamond} \mid \tilde{\mu}\alpha^{\bullet}.c\rangle$.
$\left\langle inr((\tilde{\mu}x.\langle inl(x)^{\diamond} \mid \alpha\rangle)^{\bullet})^{\diamond} \mid \alpha \right\rangle : (\vdash \alpha : P \oplus \neg^{+}P)$.
$(\mid \tilde{\mu}(x_2, x_1).\langle (x_1, x_2)^{\diamond} \mid \alpha\rangle : P_2 \otimes P_1 \vdash \alpha : P_1 \otimes P_2)$.

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

[5] The main differences with the system presented in [19] 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 [19]. Also, the present system is bilateral but limited to positive formulas on both sides, it thus corresponds to the positive fragment of the bilateral version of L_{foc} as presented in [19][long version, Appendix A].

$$
\begin{array}{lll}
\text{(control)} & \langle \mu\alpha.c \mid e\rangle \to c[e/\alpha] & \langle V^{\diamond} \mid \tilde{\mu}x.c\rangle \to c[V/x] \\
\text{(logical)} & \langle (e^{\bullet})^{\diamond} \mid \tilde{\mu}\alpha^{\bullet}.c\rangle \to c[e/\alpha] & \langle (V_1, V_2)^{\diamond} \mid \tilde{\mu}(x_1, x_2).c\rangle \to c[V_1/x_1, V_2/x_2] \\
 & \langle inl(V_1)^{\diamond} \mid \tilde{\mu}[inl(x_1).c_1 | inr(x_2).c_2]\rangle \to c_1[V_1/x_1] \quad (\text{idem } inr) & \\
\text{(commutation)} & \langle v \mid e\rangle[\sigma] \to \langle v[\sigma] \mid e[\sigma]\rangle \quad \text{etc}\ldots &
\end{array}
$$

Fig. 2. Cut eliminition in L_{foc}

PROOF. We translate the syntax given in section 3 into the focalised one. All cases are obvious except for the introduction of $\otimes$ and $\oplus$ on the right. We define $(\mu\alpha_1.c_1, \mu\alpha_2.c_2)$ 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 cuts. It also *fixes an order of evaluation*: one should read the translation of right and 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 independently in [27] and [19]). 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 [7].

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 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) [21].

Remark 3. About μ: we note that $\mu\beta.c$ is used only in a command $\langle \mu\beta.c \mid e\rangle$, and in such a context it can be expressed as $\langle (e^{\bullet})^{\diamond} \mid \tilde{\mu}\beta^{\bullet}.c\rangle$, 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 $\mu\beta.c$.

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.* As for LK, or for simply typed λ-calculus.

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

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

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

PROOF. Let π be a proof of $\Gamma \vdash \Delta$. By Propositions 1 and 2, one obtains a term denoting a focalised, cut-free proof of $\Gamma \vdash \Delta$. □

We can 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):

$$\mu\alpha.\langle v \mid \alpha\rangle = v \qquad \tilde{\mu}(x_1,x_2).\langle (x_1,x_2)^\diamond \mid e\rangle = e$$
$$\tilde{\mu}x.\langle x^\diamond \mid e\rangle = e \qquad \tilde{\mu}[inl(x_1).\langle inl(x_1)^\diamond \mid e\rangle | inr(x_2).\langle inr(x_2)^\diamond \mid e\rangle] = e$$
$$\tilde{\mu}\alpha^\bullet\langle (\alpha^\bullet)^\diamond \mid e\rangle = e$$

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.

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).

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:

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

where $\tilde{\mu}(x,\alpha^\bullet).c$ is the abbreviation used in Example 1 and where $V^\blacklozenge$ stands for $\tilde{\mu}\alpha^\bullet.\langle V^\diamond \mid \alpha\rangle$. The translation extends to (call-by-value) $\lambda\mu$-calculus [22], and factors though $\overline{\lambda}\mu\tilde{\mu}_Q$-calculus (cf. Section 2), defining $V \cdot e$ as $(V, e^\bullet)^\blacklozenge$. The translation makes also sense in the untyped setting, as the following example shows.

Example 2. Let $\Delta = \lambda x.xx$. We have $[\![\Delta\Delta]\!]^+_{\mathrm{v}} = \mu\gamma.c$, and $c \to^* c$, with

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

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. But keeping the *same* term language, we can type sequents of the form $(\ldots, \alpha : N, \ldots \vdash \ldots, x : N, \ldots)$, giving rise to a dual logic LKT, renaming the metavariables for expressions, values, and contexts as e (now contexts), E (now *covalues*, also called applicative contexts in [3]),

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:

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

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

and v (now expressions). For example the rules for left introduction of $\&$ and of right introduction for $\parr$ are as follows:

$$\frac{\Gamma\,;\, E_1 : N_1 \vdash \Delta}{\Gamma\,;\, inl(E_1) : N_1 \& N_2 \vdash \Delta} \qquad \frac{\Gamma\,;\, E_2 : N_2 \vdash \Delta}{\Gamma\,;\, inr(E_2) : N_1 \& N_2 \vdash \Delta} \qquad \frac{c : (\Gamma \vdash x_1 : N_1\ ,\ x_2 : N_2\ ,\ \Delta)}{\Gamma \vdash \tilde{\mu}(x_1, x_2).c : N_1 \parr N_2 \,|\, \Delta}$$

In what follows, it will be handier (and closer to the tradition of CBN $\lambda\mu$-calculus) to use $\tilde{x}$ (resp. $\tilde{\alpha}$) instead of a negative variable α (resp. continuation variable x).

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:

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

The translation extends to $\lambda\mu$-calculus [23] and factors though the $\overline{\lambda}\mu\tilde{\mu}_T$-calculus of [3], 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).

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. [12,14]). We write B^A for function types / intuitionistic implications. The rules of $\mathsf{L}_{\mathrm{foc}}$ are simulated by β-reductions. One may think of the source $\mathsf{L}_{\mathrm{foc}}$ terms as a description of the target ones "in direct style" (cf. [5]).

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

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

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

6 A Synthetic System

In this section we pursue two related goals.

1. We want to account for the full (or strong) focalisation (cf. [25]), 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. Example 1) 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 $\langle x^\diamond \,|\, \tilde{\mu}((x_1, x_2), (x_3, x_4)).c\rangle$, where $\tilde{\mu}((x_1, x_2), (x_3, x_4)).c$ stands for an abbreviation of either of the following two commands:

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

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

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 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:

$$\frac{}{\Xi\ ,\ x : X \vdash x : X\ ;\ \Delta} \qquad \frac{C : (\Xi\ ,\ q : P \vdash \Delta)}{\Xi \,|\, \tilde{\mu}q.C : P \vdash \Delta} \qquad \frac{C : (\Xi \vdash \alpha : P\ ,\ \Delta)}{\Xi \vdash \mu\alpha.C : P \,|\, \Delta}$$

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

$$\frac{C_1 : (\Gamma\ ,\ q_1 : P_1 \vdash \Delta) C_2 : (\Gamma\ ,\ q_2 : P_2 \vdash \Delta)}{[C_1\ ^{q_1,q_2}\ C_2] : (\Gamma\ ,\ [q_1, q_2] : P_1 \oplus P_2 \vdash \Delta)}$$

(all the other rules as in Figure 1, with Ξ in place of Γ)

Our aim now (*second step*) is to get rid of the tree structure of a command. Indeed, 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

$$[[c_{13}\ {}^{x_3,x_4}\ c_{14}]\ {}^{x_1,x_2}\ [c_{23}\ {}^{x_3,x_4}\ c_{24}]] \text{ and } [[c_{13}\ {}^{x_1,x_2}\ c_{23}]\ {}^{x_3,x_4}\ [c_{14}\ {}^{x_1,x_2}\ c_{24}]]\,.$$

To this effect, we need a last ingredient. We introduce a syntax of *patterns*, and we redefine the syntax of values, as follows:

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

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.

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

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

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:

$$\begin{array}{ll} C[(p_1,p_2)/(q_1,q_2),\sigma] \to C[p_1/q_1, p_2/q_2, \sigma] & C[\beta^\bullet/\alpha^\bullet,\sigma] \to C[\beta/\alpha,\sigma] \\ [C_1\ {}^{q_1,q_2}\ C_2][inl(p_1)/[q_1,q_2],\sigma] \to C_1[p_1/q_1,\sigma] & (\text{idem } inr) \end{array}$$

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

Example 3

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

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

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

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

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

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

$$
\begin{array}{ll}
(\tilde{\mu}^{+}) & \langle (p\,\langle \ldots, y/x, \ldots, e^{\bullet}/\alpha^{\bullet} \ldots \rangle)^{\Diamond} \,|\, \tilde{\mu}q.\{p \mapsto c_p \mid q \perp p\} \rangle \\
& \quad \rightarrow c_p\,\{\ldots, y/x, \ldots, e/\alpha, \ldots\}\} \\
(\mu) & \langle \mu\alpha.c \,|\, e \rangle \rightarrow c\{e/\alpha\}
\end{array}
$$

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

$$
\frac{\ldots \qquad \Gamma \vdash \mathcal{V}_i : P_i \,;\, \Delta \quad ((i : P_i) \in \Gamma(p,P)) \qquad \ldots}{\Gamma \vdash p\,\langle \mathcal{V}_i/i \mid i \in p \rangle : P \,;\, \Delta}
$$

$$
\frac{\ldots \qquad c_p : (\Gamma\,,\, \Gamma(p,P) \vdash \Delta) \quad (q \perp p) \qquad \ldots}{\Gamma \,|\, \tilde{\mu}q.\{p \mapsto c_p \mid q \perp p\} \vdash \Delta}
$$

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

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

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

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:

$$
[c_1\ {}^{y,\alpha^{\bullet}}\ c_2][p_1/q] = c_1\{x'/x, y'/y\} \qquad [c_1\ {}^{y,\alpha^{\bullet}}\ c_2][p_2/q] = c_2\{x'/x, \alpha'/\alpha\}
$$

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

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 4. *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.*

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 [27]: 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 5. *The typing system of* L_{synth} *is complete 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 4. In order to prove the completeness of the intermediate system, we define the following rewriting relation between sets of sequents:

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

(where α, x_1, x_2 are fresh). One proves 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 $\Xi \mid e : P \vdash \Delta$, then there exists e' such that $\Xi \mid e' : P \vdash_S \Delta$ (and similarly for expressions v),

where $\vdash_S$ (resp. $\vdash$) refers to the intermediate system (resp. to L_{foc}). The proof of 1) goes as follows. Using Lemma 1 and induction, we get simple commands c_i proving all the sequents in the normal form of $(x_1 : P_1, \ldots, x_m : P_m \vdash \Delta)$ w.r.t. the above rewriting rules. One can then assemble the c_i's to form a command C as in the statement. □

Putting together Propositions 1 and 5, we have proved that $\mathsf{L}_{\text{synth}}$ is complete with respect to LK for provability.

Remark 4. 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:

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

Compare with *Böhm trees*: $M ::= \overbrace{\lambda \boldsymbol{x}.\underbrace{P}_{c}}^{e}$ $\qquad$ $P ::= y\,\underbrace{\overbrace{M_1}^{\mathcal{V}} \ldots \overbrace{M_n}^{\mathcal{V}}}_{V}$.

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 [11], where he advocated an incremental approach to connectives, starting from a pure control kernel.

The good fit between abstract machines and our syntax $\mathsf{L}_{\mathrm{foc}}$ makes it a good candidate for being used as an intermediate language appropriate to reason about the correctness of abstract machines (see also [18]). 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 $\mathsf{L}_{\mathrm{foc}}$ that freely mixes positive and negative formulas like in Girard's LC[7].[6] Such a system is presented in the long version of [19].

We wish to thank R. Harper, H. Herbelin, and O. Laurent for helpful discussions.

References

1. Abadi, M., Cardelli, L., Curien, P.-L., Lévy, J.-J.: Explicit Substitutions. Journal of Functional Programming 1(4) (1992)
2. Andreoli, J.-M.: Logic programming with focusing proofs in linear logic. Journal of Logic and Computation 2(3), 297–347 (1992)
3. Curien, P.-L., Herbelin, H.: The duality of computation. In: Proc. Int. Conf. on Functional Programming 2000 (2000)
4. Danos, V., Joinet, J.-B., Schellinx, H.: A new deconstructive logic: linear logic. Journal of Symbolic Logic 62(3) (1997)
5. Danvy, O.: Back to direct style. Science of Computer Programming 22(3), 183–195 (1994)
6. Girard, J.-Y., Lafont, Y., Taylor, P.: Proofs and Types. Cambridge University Press, Cambridge (1989)
7. Girard, J.-Y.: A new constructive logic: classical logic. Mathematical Structures in Computer Science 1, 255–296 (1991)
8. Girard, J.-Y.: Locus solum: from the rules of logic to the logic of rules. Mathematical Structures in Computer Science 11(3), 301–506 (2001)
9. Griffin, T.: A formulae-as-types notion of control. In: Proc. Principles of Programming Languages 1990 (1990)
10. Herbelin, H.: Séquents qu'on calcule, Thèse de Doctorat, Université Paris 7 (1995), `http://pauillac.inria.fr/~herbelin`
11. Herbelin, H.: C'est maintenant qu'on calcule, au cœur de la dualité, Mémoire d'habilitation (2005) (available from cited url)
12. Krivine, J.-L.: Lambda-calcul, types et modèles, Masson (1991)
13. Krivine, J.-L.: A call-by-name lambda-calculus machine. Higher Order and Symbolic Computation 20, 199–207 (2007)
14. Lafont, Y., Reus, B., Streicher, T.: Continuation Semantics or Expressing Implication by Negation. Technical Report (1993)
15. Landin, P.: The mechanical evaluation of expressions. Computer Journal 6, 308–320 (1964)
16. Laurent, O.: Etude de la polarisation en logique. Thèse de Doctorat, Univ. Aix-Marseille II (2002)
17. Laurent, O., Quatrini, M., Tortora de Falco, L.: Polarised and focalised linear and classical proofs. Ann. of Pure and Appl. Logic 134(2-3), 217–264 (2005)
18. Levy, P.B.: Call-by-push-value. In: A functional/imperative synthesis, Semantic Structures in Computation. Springer, Heidelberg (2004)
19. Munch-Maccagnoni, G.: Focalisation and classical realisability. In: Grädel, E., Kahle, R. (eds.) CSL 2009. LNCS, vol. 5771, pp. 409–423. Springer, Heidelberg (2009), `http://perso.ens-lyon.fr/guillaume.munch/articles`

[6] See also [20] for an early analysis of the computational meaning of LC from a programming language perspective.

20. Murthy, C.: A computational analysis of Girard's translation and LC. In: Proc. LICS 1992 (1992)
21. Nipkow, T.: Orthogonal higher-order rewriting systems are confluent. In: Bezem, M., Groote, J.F. (eds.) TLCA 1993. LNCS, vol. 664, pp. 306–317. Springer, Heidelberg (1993)
22. Ong, L., Stewart, C.: A Curry-Howard foundation for functional computation with control. In: Proc. POPL '97 (1997)
23. Parigot, M.: $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In: Voronkov, A. (ed.) LPAR 1992. LNCS, vol. 624, pp. 190–201. Springer, Heidelberg (1992)
24. Plotkin, G.: Call-by-name, call-by-value and the lambda-calculus. TCS 1, 125–159 (1975)
25. Quatrini, M., Tortora de Falco, L.: Polarisation des preuves classiques et renversement. C.R.A.S.. 323(I), 113–116 (1996)
26. Wadler, P.: Call-by-value is dual to call-by-name. In: Proc. ICFP 2003 (2003)
27. Zeilberger, N.: On the unity of duality. Ann. of Pure and Appl. Logic 153(1), 66–96 (2008)