Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 332 (2017) 149–168

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

The Polarized λ-calculus

José Espírito Santo[1]

Centro de Matemática
Universidade do Minho
Braga, Portugal

Abstract

A natural deduction system isomorphic to the focused sequent calculus for polarized intuitionistic logic is proposed. The system comes with a language of proof-terms, named polarized λ-calculus, whose reduction rules express simultaneously a normalization procedure and the isomorphic copy of the cut-elimination procedure pertaining to the focused sequent calculus. Noteworthy features of this natural deduction system are: how the polarity of a connective determines the style of its elimination rule; the existence of a proof-search strategy which is equivalent to focusing in the sequent calculus; the highly-disciplined organization of the syntax - even atoms have introduction, elimination and normalization rules. The polarized λ-calculus is a programming formalism close to call-by-push-value, but justified by its proof-theoretical pedigree.

Keywords: polarized logic, focusing, bidirectional natural deduction, general elimination rule, eta-expansion

1 Introduction

Focusing and polarity are two proof-theoretical ideas that originated in linear logic but were recognized to have ample impact in proof-theory at large. Focusing [1,10,13] is a disciplined strategy for bottom-up proof-search in the sequent calculus that tries to reduce to a minimum the non-determinism inherent in that process. Polarity is the feature of a logical connective when it has a well-determined behavior in focusing proof-search. Polarized versions of intuitionistic and classical logic can be given where the logical operations have such behavior, and where even polarity *shifts* are explicit [15].

Focusing in polarized logic is a "subsuming paradigm" of proof-search, capturing several focused systems and their search strategies [10,4]. But cut-elimination in polarized logic is also a "subsuming paradigm" of functional computation, capturing the traditional dichotomy call-by-value/call-by-name through the logic qua type

[1] Email:jes@math.uminho.pt This research was financed by Portuguese Funds through FCT Fundação para a Ciência e a Tecnologia, within the Project UID/MAT/00013/2013.

http://dx.doi.org/10.1016/j.entcs.2017.04.010
1571-0661/© 2017 The Author(s). Published by Elsevier B.V.

system [15,2]. It is then quite natural to investigate the connection between the call-by-push-value (CBPV) paradigm [9] and polarized logic.

This paper will get to the point where such investigation is possible, in the intuitionistic setting. This involves several simultaneous tasks. First, we have to be ready to bridge the sequent calculus vs natural deduction divide, as the latter side is where programming notations more naturally live; second, in the spirit of the Curry-Howard correspondence, we have to equip the proof-systems with appropriate languages of proof-terms, preferably with a transparent computational interpretation; third, we have to keep track of dualities and symmetries so deeply explored by focusing, in an intuitionistic setting where they are concealed, more so in a natural deduction notation. Moreover, we need the correct methodology: we do not need to adhere to the dogma that every useful programming formalism can be reduced to proof-theory; rather, we will let proof-theory do its job, which is to produce standard systems, and after that judge the distance between such systems and those we would like to capture.

In this paper, starting from one of those standard objects, the focused sequent calculus for polarized intuitionistic logic, we propose another, the polarized λ-calculus, or $\lambda^{\pm}_{\mathsf{N}}$, under which is a natural deduction system *isomorphic* to the sequent calculus we started from. This isomorphism works at all levels: derivations, classes of derivations and their forms of judgment, reduction relations. As a result, we obtain a very interesting proof system, having a lot to say about always-hot topics in the theory of natural deduction: subformula property, proof-search, treatment of disjunction, general elimination rules [11,8,14,12]. Simultaneously, the polarized λ-calculus is an addition to the Curry-Howard correspondence: from it one can distill a variant of the CBPV calculus.

We start (section 2) with a review of focusing and polarity, as we present a sequent calculus named $\lambda^{\pm}_{\mathsf{G}}$. Section 3 is dedicated to the polarized λ-calculus $\lambda^{\pm}_{\mathsf{N}}$, with some of its properties obtained in Section 4. The closing Section 5 discusses the results.

2 A review of polarity and focusing

System $\lambda^{\pm}_{\mathsf{G}}$ has *positive* and *negative* formulas, given by the following grammar:

$$\begin{array}{rl}
\text{(Positive formulas)} & P, Q ::= a^{+} \mid {\downarrow} N \mid \perp \mid P \vee Q \\
\text{(Negative formulas)} & M, N ::= a^{-} \mid {\uparrow} P \mid P \supset N \mid N \wedge M
\end{array}$$

Here, a^{+} (resp. a^{-}) ranges over positive (resp. negative) *atoms*. Polarity *shifts* are denoted by $\downarrow$ and $\uparrow$. A *basic* positive formula B is either a positive atom, a shift $\downarrow N$, or $\perp$. A *composite negative formula* C is a negative formula that is not an atom. The following definitions are useful:

$$\text{(Formulas)}\, A ::= P \mid N \quad \text{(Right formulas)}\, R ::= P \mid a^{-} \quad \text{(Left formulas)}\, L ::= N \mid a^{+}$$

$$\frac{\Gamma \Longrightarrow t : C}{\Gamma \vdash \mathsf{dlv}(t) : C} \cdot/Jump$$

$$\frac{\Gamma \vdash [V : P]}{\Gamma \vdash \mathsf{ret}(V) : P} \cdot/Focus_R \qquad \frac{\Gamma, x : N[S : N] \vdash A}{\Gamma, x : N \vdash \mathsf{coret}(x, S) : A} \cdot/Focus_L$$

$$\frac{\Gamma \Longrightarrow t : N \quad \Gamma[S : N] \vdash A}{\Gamma \vdash \langle t|S\rangle : A} Cut^-/\cdot \qquad \frac{\Gamma \vdash [V : P] \quad \Gamma|p : P \Longrightarrow A}{\Gamma \vdash \langle V|p\rangle : A} Cut^+/\cdot$$

Fig. 1. Typing rules of $\lambda^{\pm}_{\mathsf{G}}$ - rules for stable sequents $\Gamma \vdash e : A$

The inference/typing rules of $\lambda^{\pm}_{\mathsf{G}}$, given in Figs. 1, 2 and 3, handle the following sequents:

$$\Gamma \vdash [V : P] \qquad \Gamma \Longrightarrow t : N \qquad \Gamma[S : N] \vdash A \qquad \Gamma|p : P \Longrightarrow A \qquad \Gamma \vdash e : A \ .$$

A context Γ is a set of declarations $x : L$ (hence either $x : N$ or $z : a^+$), where each variable x is declared at most once. Inference rules are doubly tagged as $[tag1/tag2]$. The first tag gives the usual name to the rule according to the top-down reading of it; the second tag gives a name according to the action the rule produces when applied bottom-up in the process of proof-search. If no tag is appropriate, we put $\cdot$.

Proof-search. The first thing to do in order to understand this system is to strip all the term annotations, including the variables in declarations. Γ becomes temporarily a multisets of left formulas and we get five forms of sequents:

- $\Gamma \vdash A$ - stable sequent
- $\Gamma \vdash [P]$ - focus on the positive P in the r.h.s. of the sequent.
- $\Gamma \Longrightarrow N$ - invert the negative N in the r.h.s. of the sequent.
- $\Gamma[N] \vdash A$ - focus on the negative N in the l.h.s. of the sequent.
- $\Gamma|P \Longrightarrow A$ - invert the positive P in the l.h.s. of the sequent.

Proof-search is defined in the subsystem where the cut-rules are omitted and proceeds as follows. Given a stable sequent $\Gamma \vdash A$, an external decision is required, namely the choice of which formula of the sequent to focus on: either the positive P (if $A = P$), or a negative $N \in \Gamma$ (if a negative exists in Γ).

- In the first case, the decomposition of P will produce either a positive atom or a negative M after the *blur* of the focus. In the former sub-case, one *accepts* the sequent, if the produced atom is in Γ, otherwise the search fails. In the latter sub-case, the process proceeds with the inversion of M. Right inversion *collects* on the r.h.s. either a negative atom, or a positive, which makes the sequent stable

$$\frac{}{\Gamma, z : a^+ \vdash [z : a^+]}\ Atom_R^+/Accept^+ \qquad \frac{\Gamma \Longrightarrow t : N}{\Gamma \vdash [\mathsf{thunk}(t) :\downarrow N]}\ \downarrow_R /Blur_R$$

$$\frac{\Gamma \vdash [V : P_i]}{\Gamma \vdash [\mathsf{in}_i(V) : P_1 \vee P_2]}\ \vee_R/\cdot\ (i = 1, 2)$$

$$\frac{}{\Gamma[\mathsf{nil} : a^-] \vdash a^-}\ Atom_L^-/Accept^- \qquad \frac{\Gamma|p : P \Longrightarrow A}{\Gamma[\mathsf{cothunk}(p) :\uparrow P] \vdash A}\ \uparrow_L /Blur_L$$

$$\frac{\Gamma \vdash [V : P] \quad \Gamma[S : N] \vdash A}{\Gamma[V :: S : P \supset N] \vdash A}\ \supset_L /\cdot \qquad \frac{\Gamma[S : N_i] \vdash A}{\Gamma[\mathsf{i} :: S : N_1 \wedge N_2] \vdash A}\ \wedge_L/\cdot\ (i = 1, 2)$$

Fig. 2. Typing rules of $\lambda^{\pm}_{\mathsf{G}}$ - rules for focusing sequents $\Gamma \vdash [V : P]$ and $\Gamma[S : N] \vdash A$

$$\frac{\Gamma \vdash e : a^-}{\Gamma \Longrightarrow \ulcorner e \urcorner : a^-}\ Atom_R^-/Collect_R^- \qquad \frac{\Gamma \vdash e : P}{\Gamma \Longrightarrow \lceil e \rceil :\uparrow P}\ \uparrow_R /Collect_R^+$$

$$\frac{\Gamma|p : P \Longrightarrow N}{\Gamma \Longrightarrow \lambda p : P \supset N}\ \supset_R /\cdot \qquad \frac{\Gamma \Longrightarrow t_1 : N_1 \quad \Gamma \Longrightarrow t_2 : N_2}{\Gamma \Longrightarrow \langle t_1, t_2 \rangle : N_1 \wedge N_2}\ \wedge_R/\cdot$$

$$\frac{\Gamma, z : a^+ \vdash e : A}{\Gamma|z.e : a^+ \Longrightarrow A}\ Atom_L^+/Collect_L^+ \qquad \frac{\Gamma, x : N \vdash e : A}{\Gamma|x.e :\downarrow N \Longrightarrow A}\ \downarrow_L /Collect_L^-$$

$$\frac{}{\Gamma|\mathsf{abort} :\perp \Longrightarrow A}\ \perp_L /\cdot \qquad \frac{\Gamma|p_1 : P_1 \Longrightarrow A \quad \Gamma|p_2 : P_2 \Longrightarrow A}{\Gamma|[p_1, p_2] : P_1 \vee P_2 \Longrightarrow A}\ \vee_L/\cdot$$

Fig. 3. Typing rules of $\lambda^{\pm}_{\mathsf{G}}$ - rules for inversion sequents $\Gamma \Longrightarrow t : N$ and $\Gamma|p : P \Longrightarrow A$

again.

- In the second case, the decomposition of N will produce either a negative atom or a positive Q after the *blur* of the focus. In the former sub-case, one *accepts*

$$
\begin{array}{rrcl}
\text{(Values)} & V & ::= & z \,|\, \mathsf{thunk}(t) \,|\, \mathsf{in}_1(V) \,|\, \mathsf{in}_2(V) \\
\text{(Terms)} & t & ::= & \ulcorner e \urcorner \,|\, \lceil e \rceil \,|\, \lambda p \,|\, \langle t_1, t_2 \rangle \\
\text{(Co}-\text{values, aka spines)} & S & ::= & \mathsf{nil} \,|\, \mathsf{cothunk}(p) \,|\, V :: S \,|\, 1 :: S \,|\, 2 :: S \\
\text{(Co}-\text{terms)} & p & ::= & z.e \,|\, x.e \,|\, [p_1, p_2] \,|\, \mathsf{abort} \\
\text{(Expressions)} & e & ::= & \mathsf{dlv}(t) \,|\, \mathsf{ret}(V) \,|\, \mathsf{coret}(x, S) \,|\, \langle t | S \rangle \,|\, \langle V | p \rangle
\end{array}
$$

Fig. 4. Proof-terms for $\lambda^{\pm}_{\mathsf{G}}$

the sequent, if the produced atom is R, otherwise the search fails. In the latter sub-case, the process proceeds with the inversion of Q. Left inversion either stops if $\perp_L$ is hit, or *collects* on the l.h.s. either a positive atom, or a negative, which makes the sequent stable again.

This pleasing symmetric picture is slightly disturbed by the fact that the r.h.s. formula A of the initial stable sequent may be a composite negative formula C. In this case, a third possibility for the search procedure exists: to *jump* directly to the inversion of C.

Why can't the r.h.s. formula in $\Gamma \vdash A$, in $\Gamma[N] \vdash A$, and in $\Gamma|P \Longrightarrow A$ be restricted to a right formula R? The reason is inference rule $\supset_R$ in Fig. 3: the inversion sequent in the premiss has a negative r.h.s. formula, hence the stable sequents collected at the end of the left inversion phase may have a negative formula on the r.h.s., hence the same is true of left focusing sequents due to rule $Focus_L$ in Fig. 1 (for if the r.h.s. formula of sequents $\Gamma[N] \vdash A$ were restricted to R, we would be forced to jump whenever a stable sequent had a formula C in the r.h.s., even if there were a N in the l.h.s. to focus on).

A language of proof-terms. The syntax of the proof-terms of $\lambda^{\pm}_{\mathsf{G}}$ is given in Fig.4.

There are 5 syntactic classes because there are 5 kinds of sequents in the logical system. We assume two denumerable, disjoint sets of *negative variables* and *positive variables*, ranged over by x and z, respectively. In the case of $\Gamma[S : N] \vdash R$, $\Gamma|p : P \Longrightarrow R$ or $\Gamma \vdash e : R$, we may think of R as a co-context (=r.h.s. context) with a single declaration, either of the *default positive co-variable*, say $\star$, with positive type, or of the *default negative co-variable* with atomic negative type. After comparing the typing rules $Atom^{+}_{R}$ and $Atom^{-}_{L}$, one has to recognize nil as the default negative co-variable, whereas $\star$ is nowhere written. We may think that $\star$ has a unique, implicit occurrence in each positive S, p, or e, while it has none in V, or t, or negative e.

We should look again at the system in Figs. 1, 2 and 3, now as a typing system, with all the term annotations put back. We think of values V and terms t as typed on the right, with a positive and a negative type respectively; and think of co-values S and co-terms p as *co-typed* on the left, with a negative and positive type,

$$\frac{\Gamma \Longrightarrow t : N \quad \Gamma, x : N \vdash e : A}{\Gamma \vdash [t/x]e : A} \qquad \frac{\Gamma \vdash e : P \quad \Gamma|p : P \Longrightarrow A}{\Gamma \vdash e[\star\backslash p] : A}$$

$$\frac{\Gamma|p : P \Longrightarrow N \quad \Gamma[S : N] \vdash A}{\Gamma|p@S : P \Longrightarrow A} \qquad \frac{\Gamma \vdash e : N \quad \Gamma[S : N] \vdash A}{\Gamma \vdash e@S : A}$$

$$\frac{\Gamma[S' : A'] \vdash N \quad \Gamma[S : N] \vdash A}{\Gamma[S'@S : A'] \vdash A}$$

Fig. 5. Admissible typing rules for the syntactic operations of $\lambda^{\pm}_{\mathsf{G}}$

respectively. Expressions annotate stable sequents.

The duality between syntactic classes is also seen at their particular constructions. The values z and $\mathsf{thunk}(t)$ are dual to nil and $\mathsf{cothunk}(p)$, respectively. If we write $\ulcorner e \urcorner$ and $\lceil e \rceil$ as the co-bindings $e.\mathsf{nil}$ and $e.\star$, one sees the duality with the bindings $z.e$ and $x.e$, respectively.

The constructors for the three connectives $\wedge$, $\vee$, and $\supset$ are pairing, injection and λ-abstraction, the latter in the unusual form λp, where p is a co-pairing of a number of "bodies" of the forms $z.e$, $x.e$ or abort. Co-pairing is the destructor for $\vee$, while the negatives $\wedge$ and $\supset$ have a generic form of destruction, namely pushing the spine (with either a value V or a projection symbol 1, 2).

If we disregard the delivered term $\mathsf{dlv}(t)$ (which expresses the jump facility introduced in the proof-system), the four forms of an expression e come as two pairs of dual constructions. The duality between return and co-return is enhanced if we think of the former as $\mathsf{ret}(V, \star)$. There is a positive cut $\langle V|p\rangle$ and a negative cut $\langle t|S\rangle$: the polarity of a cut is that of its cut-formula.

Derived syntax. There is an operation of *negative substitution* $[t/x]e$, in whose recursive definition the critical equation is $[t/x]\mathsf{coret}(x, S) = \langle t|S'\rangle$, where $S' = [t/x]S$. Dually, there is a *positive substitution* $e[\star\backslash p]$, in whose recursive definition the critical equation is $\mathsf{ret}(V)[\star\backslash p] = \langle V|p\rangle$. The admissible typing rules are in Fig. 5.

We need a third operation $p@S$. If $S = \mathsf{nil}$, let $p@S = p$, $e@S = e$, and $S'@S = S'$; otherwise, the co-term $p@S$, the expression $e@S$, and the spine $S'@S$

$$
\begin{array}{ll}
\boxed{\supset} & \langle \lambda p | V :: S \rangle \to \langle V | p@S \rangle \\
\boxed{\wedge} & \langle \langle t_1, t_2 \rangle | \mathsf{i} :: S \rangle \to \langle t_i | S \rangle \ (i = 1, 2) \\
\boxed{\uparrow} & \langle \lceil e \rceil | \mathsf{cothunk}(p) \rangle \to e[\star \backslash p] \\
\boxed{a^-} & \langle \ulcorner e \urcorner | \mathsf{nil} \rangle \to e \\
\boxed{\vee} & \langle \mathsf{in}_i(V) | [p_1, p_2] \rangle \to \langle V | p_i \rangle \ (i = 1, 2) \\
\boxed{\downarrow} & \langle \mathsf{thunk}(t) | x.e \rangle \to [t/x]e \\
\boxed{a^+} & \langle z | z'.e \rangle \to [z/z']e
\end{array}
$$

Fig. 6. Cut-elimination in $\lambda_{\mathsf{G}}^{\pm}$

are defined by simultaneous recursion on p and e and S':

$$
\begin{array}{rclrcl}
\mathsf{abort}@S & = & \mathsf{abort} & (\mathsf{dlv}(t))@S & = & \langle t | S \rangle \\
[p_1, p_2]@S & = & [p_1@S, p_2@S] & \langle V | p \rangle @S & = & \langle V | p@S \rangle \\
(x.e)@S & = & x.(e@S) & \langle t | S' \rangle @S & = & \langle t | S'@S \rangle \\
(z.e)@S & = & z.(e@S) & \mathsf{coret}(x, S')@S & = & \mathsf{coret}(x, S'@S)
\end{array}
$$

$$
\begin{array}{rcl}
\mathsf{nil}@S & = & S \\
\mathsf{cothunk}(p)@S & = & \mathsf{cothunk}(p@S) \\
(V :: S')@S & = & V :: (S'@S) \\
(\mathsf{i} :: S')@S & = & \mathsf{i} :: (S'@S)
\end{array}
$$

One inspects p or e or S', searching for a delivered term t that will create the cut $\langle t | S \rangle$.

Polarized cut-elimination. The cut-elimination rules are given in Fig.6. These rules are relations on expressions and generate, by compatible closure, one-step reduction relations denoted $\to$. The fist four rules reduce negative cuts, the last three positive cuts. Notice that the reduction of a negative cut may generate a positive cut - this is already visible in the first rule. Every redex is a cut. In a typed expression, every cut is a redex, as typing forces the cuts to have the form of one of the displayed redexes. For instance, in a typed cut $\langle \lambda p | S \rangle$, S has to have the form $V :: S'$, since this is the only form of S which is co-typed with an implication (in the left focus). So, a typed expression is cut-free iff is irreducible. Moreover, every cut is principal, as the cut-formula is introduced in both premisses, even when the cut-formula is atomic (a consequence of having inference/typing rules for introducing

$$\frac{\Gamma \Longrightarrow t : C}{\Gamma \vdash \mathsf{dlv}(t) : C} \cdot/Jump \qquad \frac{\Gamma \vdash [V : P]}{\Gamma \vdash \mathsf{ret}(V) : P} Id^{+}/Focus_U$$

$$\frac{\Gamma \rhd H : a^{-}}{\Gamma \vdash \mathsf{ap}(H) : a^{-}} Atom_E^{-}/Accept^{-}$$

$$\frac{\Gamma \rhd H : \uparrow P \quad P = B_1 \vee \cdots \vee B_n \quad (\Gamma | b_i : B_i \longrightarrow A)_{i=1,\cdots,n}}{\Gamma \vdash \mathsf{case\,ap}(H, b_1, \cdots, b_n) : A} \uparrow_E /BIC_D$$

$$\frac{\Gamma \vdash [V : P] \quad P = B_1 \vee \cdots \vee B_n \quad (\Gamma | b_i : B_i \longrightarrow A)_{i=1,\cdots,n}}{\Gamma \vdash \mathsf{case\,val}(V, b_1, \cdots, b_n) : A} +_E/\cdot$$

Fig. 7. Typing rules of $\lambda_{\mathsf{N}}^{\pm}$ - rules for stable sequents $\Gamma \vdash e : A$

atoms both in the l.h.s. and r.h.s. of sequents).

3 The polarized λ-calculus

A natural deduction system for polarized intuitionistic logic. The inference/typing rules of $\lambda_{\mathsf{N}}^{\pm}$ are given in Figs. 7, 8 and 9. They handle the following sequents:

$$\Gamma \vdash [V : P] \qquad \Gamma \Longrightarrow t : N \qquad \Gamma \rhd H : N \qquad \Gamma \vdash e : A \ .$$

A context Γ is again a set of declarations $x : L$ (hence either $x : N$ or $z : a^{+}$), where each variable x is declared at most once. The inference rules of $\lambda_{\mathsf{N}}^{\pm}$ are doubly tagged as $[tag1/tag2]$. The first tag gives a name to the rule according to the top-down reading of it; the second tag gives a name according to the action the rule produces in the process of proof-search to be introduced below. If no tag is appropriate, we put $\cdot$.

The first thing to do in order to understand this system is to concentrate on the logical side, omitting all term annotations, including the variables in declarations. Contexts Γ become temporarily multisets of left formulas (N or a^{+}).

Recall that B ranges over basic positive formulas, that is, formulas of the forms a^{+}, $\downarrow N$ or $\bot$. Every positive formula is thus of the form $B_1 \vee \cdots \vee B_n$. Some inference rules have premisses of the form $\Gamma | B \longrightarrow A$. This is a derived form of sequent which is defined by case analysis of B:

$$\Gamma | a^{+} \longrightarrow A := \Gamma, a^{+} \vdash A \qquad \Gamma | \downarrow N \longrightarrow A := \Gamma, N \vdash A \qquad \Gamma | \bot \longrightarrow A := true \tag{1}$$

$$\frac{}{\Gamma, z : a^+ \vdash [z : a^+]}\ Atom_I^+/Accept^+ \qquad \frac{\Gamma \Longrightarrow t : N}{\Gamma \vdash [\mathsf{thunk}(t) :\downarrow N]}\ \downarrow_I /Blur_U$$

$$\frac{\Gamma \vdash [V : B_k]}{\Gamma \vdash [\mathsf{in}_k^n(V) : B_1 \vee \cdots \vee B_n]}\ \vee_I/\cdot\ (n \geq 2, 1 \leq k \leq n)$$

$$\frac{}{\Gamma, x : N \rhd x : N}\ Id^-/Focus_D \qquad \frac{\Gamma \Longrightarrow t : N}{\Gamma \rhd \mathsf{hd}(t) : N}\ -_I/\cdot$$

$$\frac{\Gamma \rhd H : P \supset N \quad \Gamma \vdash [V : P]}{\Gamma \rhd HV : N}\ \supset_E /\cdot \qquad \frac{\Gamma \rhd H : N_1 \wedge N_2}{\Gamma \rhd H\mathsf{i} : N_i}\ \wedge_E/\cdot\ (i = 1, 2)$$

Fig. 8. Typing rules of $\lambda_{\mathsf{N}}^{\pm}$ - rules for focusing sequents $\Gamma \vdash [V : P]$ and $\Gamma \rhd H : N$

$$\frac{\Gamma \vdash e : a^-}{\Gamma \Longrightarrow \ulcorner e \urcorner : a^-}\ Atom_I^-/Collect_U^- \qquad \frac{\Gamma \vdash e : P}{\Gamma \Longrightarrow \lceil e \rceil :\uparrow P}\ \uparrow_I /Collect_U^+$$

$$\frac{P = B_1 \vee \cdots \vee B_n \quad (\Gamma | b_i : B_i \longrightarrow N)_{i=1,\cdots,n}}{\Gamma \Longrightarrow \lambda(b_1, \cdots, b_n) : P \supset N}\ \supset_I /\cdot$$

$$\frac{\Gamma \Longrightarrow t_1 : N_1 \quad \Gamma \Longrightarrow t_2 : N_2}{\Gamma \Longrightarrow \langle t_1, t_2 \rangle : N_1 \wedge N_2}\ \wedge_I/\cdot$$

Fig. 9. Typing rules of $\lambda_{\mathsf{N}}^{\pm}$ - rules for inversion sequents $\Gamma \Longrightarrow t : N$

The third equation of (1) means that we may erase premisses of the form $\Gamma| \perp \longrightarrow A$. The first (resp. second) equation of (1) hides the elimination of positive atoms (resp. elimination of $\downarrow$). For example, the following is an instance of rule $+_E$:

$$\frac{\Gamma \vdash [a^+ \vee \perp \vee \downarrow N] \quad \Gamma, a^+ \vdash A \quad \Gamma, N \vdash A}{\Gamma \vdash A}$$

In order to understand the top-down reading of the inference rules, let us forget temporarily the distinction between the various forms of sequents. The connectives $\wedge$ and $\supset$ have primitive introduction and elimination rules. The rules for $\wedge$ and the elimination rule for $\supset$ are the easiest. Notice that the introduction rule for $\supset$ has this familiar particular case:

$$\frac{\Gamma, M \vdash N}{\Gamma \Longrightarrow \downarrow M \supset N}$$

Also familiar is the introduction rule for $\vee$.

The up shift $\uparrow$ also has an introduction and an elimination rule, but the latter is complex, as it may involve the elimination of $\vee$ and of basic atomic positive formulas through the definition (1). This rule is similar to $+_E$, which is a *generic elimination rule for positive formulas.* Dually, there is a *generic introduction rule for negative formulas* $-_I$: while in $+_E$ we eliminate a focused positive, in $-_I$ we conclude a focused negative, that starts a downward focusing phase (see below for proof search in this system). The introduction rule for the down shift $\downarrow$ is primitive, while, again, its elimination rule is hidden in (1).

Since Γ contains formulas of the forms a^+ or N, one has the axioms $Atom_I^+$ and Id^-. The latter starts a negative downward focus from an assumption, while its dual Id^+ starts a positive upward focus from a conclusion. On the other hand, $Atom_I^+$ belongs to a set of four rules for atoms, one of which is hidden in (1). As any former of negative formulas, negative atoms have an introduction rule $Atom_I^-$; as the other base case of a negative formula ($\uparrow P$), negative atoms have an elimination rule that ends a downward focusing phase.

There remains the single rule without first tag: as in sequent calculus, it is a jump facility only explainable through proof-search.

Normality. In a derivation of a stable sequent, a positive formula P is *maximal* (=conclusion of an introduction and main premiss of an elimination) iff P is the right focus of the main premiss of an instance of $+_E$ (because a derivation of $\Gamma \vdash [P]$ necessarily ends with an introduction, and no other elimination rule has a positive formula in the main premiss); and a negative formula N is maximal iff it is the r.h.s. formula of an instance of $-_I$ (because every main premiss of an elimination has the form $\Gamma \rhd N$; no rule for deriving this kind of sequent, other than $-_I$, is an introduction; and all rules for deriving the premiss of $-_I$ are introductions). These observation justify the following definition.

Definition 3.1 A derivation in $\lambda_{\mathsf{N}}^{\pm}$ is said to be *normal* if it contains no occurrence of rules $-_I$ or $+_E$ (the generic rules).

Proposition 3.2 (Subformula property) (i) *In a normal derivation of* $\Gamma \vdash [P]$ *(resp.* $\Gamma \Longrightarrow N$, $\Gamma \vdash A$*) every formula is a subformula of* P *(resp.* N, A*) or of some formula in* Γ*.*

(ii) *In a normal derivation of* $\Gamma \rhd N$*, every formula, including* N*, is a subformula of some formula in* Γ*.*

Proof. By simultaneous induction on $\Gamma \vdash [P]$, $\Gamma \Longrightarrow N$, $\Gamma \vdash A$, and $\Gamma \rhd N$. □

Proof search. We define a proof-search procedure for $\lambda_{\mathsf{N}}^{\pm}$ which explains the difference between the four forms of sequents:

- $\Gamma \vdash A$ - stable sequent

$$\begin{array}{rrcl}
\text{(Values)} & V & ::= & z \mid \mathsf{thunk}(t) \mid \mathsf{in}^n_k(V) \\
\text{(Terms)} & t & ::= & \ulcorner e \urcorner \mid \lceil e \rceil \mid \lambda \overrightarrow{b} \mid \langle t_1, t_2 \rangle \\
\text{(Heads, hole expressions)} & H & ::= & x \mid HV \mid H1 \mid H2 \mid \mathsf{hd}(t) \\
\text{(Expressions)} & e & ::= & \mathsf{dlv}(t) \mid \mathsf{ret}(V) \mid \mathsf{ap}(H) \mid \mathsf{case\,ap}(H, \overrightarrow{b}) \mid \mathsf{case\,val}(V, \overrightarrow{b})
\end{array}$$

Fig. 10. Proof-terms for $\lambda^{\pm}_{\mathsf{N}}$

- $\Gamma \vdash [P]$ - focus up on the positive P (in the r.h.s. of the sequent).
- $\Gamma \Longrightarrow N$ - invert up the negative N (in the r.h.s. of the sequent).
- $\Gamma \rhd N$ - focus down on the negative N (in the r.h.s. of the sequent).

As in sequent calculus, proof-search here is organized into phases of focusing and inversion, but all the "action" now happens in the r.h.s. of sequents. We now have *upward* (rather than right) and *downward* (rather than left) focusing and inversion. In the upward (resp. downward) phase, inference rules are applied bottom-up (resp. top-down). Proof-search is defined over normal derivations.

Given a stable sequent $\Gamma \vdash A$, one has to chose either to start upward focusing on the positive P (if $A = P$), or downward focusing on a negative $N \in \Gamma$ (if one such N exists).

- In the first case, one proceeds exactly as in the right focusing phase of proof-search in sequent calculus.
- In the second case, we suspend the bottom-up development of the given stable sequent, and turn to a phase of top-down application of inference rules. A negative formula N from Γ is chosen by rule $Focus_D$ to start the downward focusing. The decomposition of N through the elimination rules for $\supset$ and $\wedge$ will produce either a negative atom or some $\uparrow Q$. In the former sub-case, one *accepts* the given stable sequent, if the produced atom is R, otherwise the search fails. In the latter sub-case, we return to the given stable sequent and proceed with the bottom-up application of BIC_D, which stands for *downward blur-invert-collect*. By this single rule, the downward focus $\uparrow Q$ is blurred, the positive Q inverted, positive atoms and negatives at the bottom of Q collected, and, for each of these, a new stable sequent created.

As in sequent calculus, this proof-search procedure is slightly extended: if A is some composite negative C, then the process may *jump* directly to the inversion of C.

A language of proof-terms. The syntax of the proof-terms of $\lambda^{\pm}_{\mathsf{G}}$ is given in Fig.10. There are 4 syntactic classes in this language because there are 4 kinds of sequents in the logical system. One should now return to the system in Figs.7, 8 and 9 and put all the term annotations back. We have a typing system for this term language.

In $\lambda^{\pm}_{\mathsf{N}}$, *basic co-terms* b form a derived syntactic class defined by this grammar:

$$b ::= z.e \,|\, x.e \,|\, \mathsf{abort}$$

Accordingly, there is a derived form of sequents $\Gamma|b : B \longrightarrow A$ defined thus:

$$\begin{aligned} \Gamma|z.e : a^{+} \longrightarrow A &:= \Gamma, z : a^{+} \vdash e : A \\ \Gamma|x.e :\downarrow N \longrightarrow A &:= \Gamma, x : N \vdash e : A \\ \Gamma|\mathsf{abort} :\perp \longrightarrow A &:= \mathit{true} \end{aligned} \tag{2}$$

A *co-term* in $\lambda^{\pm}_{\mathsf{N}}$ is a vector $\overrightarrow{b} = b_1, \cdots, b_n$ of basic co-terms. Notice that in $\lambda^{\pm}_{\mathsf{N}}$ co-terms are not proof-terms *per se* (contrary to $\lambda^{\pm}_{\mathsf{G}}$). For instance, abort is just a symbol that may be used in the formation of λ-abstraction and case-expressions.

Relatively to the language of proof-terms of $\lambda^{\pm}_{\mathsf{G}}$, the novelty is the class of *heads*, which are series of "applications" starting from a negative variable x or a *head term*. Here "application" comprehends the usual application HV and also projections $H\mathsf{i}$, as the notation suggests. Hence $\mathsf{ap}(H)$ (which we may type when H has atomic type) is called "applicative" expression. If H has type $\uparrow P$, we may type a *negative case-expression* $\mathsf{case\,ap}(H, \overrightarrow{b})$. The other form of case expression is said *positive*.

Definition 3.3 An expression of $\lambda^{\pm}_{\mathsf{N}}$ is *normal* if it contains no occurrence of hd or case val.

Derived syntax. Applicative expressions and negative case-expressions may be uniformly given as $[H]\mathcal{S}$, denoting the result of filling H in the hole of $\mathcal{S}$, where $\mathcal{S}$ is a context (an expression with a hole) with two possible forms[2]:

$$\mathsf{ap}([\_]W_1 \cdots W_m) \qquad\qquad \mathsf{case\,ap}([\_]W_1 \cdots W_m, \overrightarrow{b}) \ .$$

Here each W_i is either a value V or a projection symbol 1, 2. Notice the hole of $\mathcal{S}$ expects an H, and this is why heads may also be called hole expressions. Such contexts are the *spines* of $\lambda^{\pm}_{\mathsf{N}}$ and are inductively generated by

$$\mathcal{S} ::= \mathsf{ap}([\_]) \,|\, \mathsf{case\,ap}([\_], \overrightarrow{b}) \,|\, [[\_]V]\mathcal{S} \,|\, [[\_]1]\mathcal{S} \,|\, [[\_]2]\mathcal{S} \tag{3}$$

So, in order to generate spines[3] we fill holes of spines with $[\_]V$ or $[\_]\mathsf{i}$.

In addition to hole filling in spines, there are two other, somewhat relaxed, concepts of hole filling. The first is this: we define what it means to fill e of non-atomic negative type C in the hole of $\mathcal{S}$, producing an expression denoted $[\![e]\!]\mathcal{S}$, and

[2] Notice we are diverging from the usual notation for hole-filling, which would be $\mathcal{S}[H]$. A justification for the adopted notation is that the hole of $\mathcal{S}$ is at the left end of the expression.

[3] The justification for the terminology "spines" will come later: the maps that connect $\lambda^{\pm}_{\mathsf{G}}$ and $\lambda^{\pm}_{\mathsf{N}}$ will relate precisely spines in the former with "spines" in the latter.

$$\frac{\Gamma \vdash e : N \quad \Gamma[\mathcal{S} : N] \longrightarrow A}{\Gamma \vdash [\![e]\!]\mathcal{S} : A} \qquad \frac{\Gamma | b : B \longrightarrow N \quad \Gamma[\mathcal{S} : N] \vdash A}{\Gamma | b@\mathcal{S} : B \longrightarrow A}$$

$$\frac{\Gamma \vdash e : P \quad P = B_1 \vee \cdots \vee B_n \quad (\Gamma | b_i : B_i \longrightarrow A)_{i=1,\cdots,n}}{\Gamma \vdash \llbracket e \rrbracket (b_1, \cdots, b_n) : A}$$

$$\frac{\Gamma | b : Q \longrightarrow P \quad P = B_1 \vee \cdots \vee B_n \quad (\Gamma | b_i : B_i \longrightarrow A)_{i=1,\cdots,n}}{\Gamma | b@' \overrightarrow{b} : Q \longrightarrow A}$$

$$\frac{\Gamma \Longrightarrow t : N \quad \Gamma, x : N \vdash e : A}{\Gamma \vdash [t/x]e : A}$$

Fig. 11. Admissible typing rules for the syntactic operations of $\lambda^{\pm}_{\mathsf{N}}$

simultaneously we define the basic co-term $b@\mathcal{S}$:

$$\begin{array}{rclrcl}
[\![\mathsf{dlv}(t)]\!]\mathcal{S} & = & [\mathsf{hd}(t)]\mathcal{S} & \mathsf{abort}@\mathcal{S} & = & \mathsf{abort} \\
[\![\mathsf{case\,val}(V, \overrightarrow{b})]\!]\mathcal{S} & = & \mathsf{case\,val}(V, (\overrightarrow{b})@\mathcal{S}) & (x.e)@\mathcal{S} & = & x.([\![e]\!]\mathcal{S}) \\
[\![\mathsf{case\,ap}(H, \overrightarrow{b})]\!]\mathcal{S} & = & \mathsf{case\,ap}(H, (\overrightarrow{b})@\mathcal{S}) & (z.e)@\mathcal{S} & = & z.([\![e]\!]\mathcal{S})
\end{array}$$

where $(b_1, \cdots, b_n)@\mathcal{S} = b_1@\mathcal{S}, \cdots, b_n@\mathcal{S}$. In the definition of $[\![e]\!]\mathcal{S}$ one searches e to find a delivered term $\mathsf{dlv}(t)$, and proceed with the ordinary filling of $\mathsf{hd}(t)$ in $\mathcal{S}$. We extend the operations to the case where a^- is the type of e and the co-type of $\mathcal{S}$, by putting: $[\![e]\!]\mathsf{ap}([\_]) = e$[4] and $b@\mathsf{ap}([\_]) = b$. The admissible typing rules are in Fig. 11.

The second relaxed concept of hole filling is $\llbracket e \rrbracket \overrightarrow{b}$, where we regard $\overrightarrow{b}$ as the context $\mathsf{case\,val}([\_], \overrightarrow{b})$, for which it is defined what it means to fill a value V. We define the expression $\llbracket e \rrbracket \overrightarrow{b}$ together with the basic co-term $b@' \overrightarrow{b}$:

$$\begin{array}{rclrcl}
\llbracket \mathsf{ret}(V) \rrbracket \overrightarrow{b} & = & \mathsf{case\,val}(V, \overrightarrow{b}) & \mathsf{abort}@' \overrightarrow{b} & = & \mathsf{abort} \\
\llbracket \mathsf{case\,ap}(H, \overrightarrow{b}') \rrbracket \overrightarrow{b} & = & \mathsf{case\,ap}(H, (\overrightarrow{b}')@' \overrightarrow{b}) & (z.e)@' \overrightarrow{b} & = & z.(\llbracket e \rrbracket \overrightarrow{b}) \\
\llbracket \mathsf{case\,val}(V, \overrightarrow{b}') \rrbracket \overrightarrow{b} & = & \mathsf{case\,val}(V, (\overrightarrow{b}')@' \overrightarrow{b}) & (x.e)@' \overrightarrow{b} & = & x.(\llbracket e \rrbracket \overrightarrow{b})
\end{array}$$

where $(b'_1, \cdots, b'_n)@' \overrightarrow{b} = b'_1@' \overrightarrow{b}, \cdots, b'_n@' \overrightarrow{b}$. In the definition of $\llbracket e \rrbracket \overrightarrow{b}$ one searches e to find a returned value $\mathsf{ret}(V)$, and proceed with the ordinary filling of V

[4] We might have put $[\![e]\!]\mathsf{ap}([\_]) = \mathsf{ap}(\mathsf{hd}(\ulcorner e \urcorner))$ but we will see that e is a reduct of $\mathsf{ap}(\mathsf{hd}(\ulcorner e \urcorner))$.

$$\begin{array}{lrcll}
\boxed{\supset} & [\big(\mathsf{hd}(\lambda \overrightarrow{b})\big)\, V]\mathcal{S} & \to & \mathsf{case\,val}(V, \overrightarrow{b}@\mathcal{S}) & \\
\boxed{\wedge} & \big(\mathsf{hd}(\langle t_1, t_2\rangle)\big)\mathsf{i} & \to & \mathsf{hd}(t_i) & (i = 1, 2) \\
\boxed{\uparrow} & \mathsf{case\,ap}(\mathsf{hd}(\lceil e \rceil), \overrightarrow{b}) & \to & [\![e]\!]\, \overrightarrow{b} & \\
\boxed{a^-} & \mathsf{ap}(\mathsf{hd}(\ulcorner e \urcorner)) & \to & e & \\
\boxed{\vee} & \mathsf{case\,val}(\mathsf{in}_k^n(V), b_1, \cdots, b_n) & \to & \mathsf{case\,val}(V, b_k) & (n \geq 2, 1 \leq k \leq n) \\
\boxed{\downarrow} & \mathsf{case\,val}(\mathsf{thunk}(t), x.e) & \to & [t/x]e & \\
\boxed{a^+} & \mathsf{case\,val}(z, z'.e) & \to & [z/z']e &
\end{array}$$

Fig. 12. Normalization in $\lambda^{\pm}_{\mathsf{N}}$

in $\mathsf{case\,val}([\_], \overrightarrow{b})$. The case missing cases of e do not arise since the hole of $\mathsf{case\,val}([\_], \overrightarrow{b})$ has positive type. Again the admissible typing rules are in Fig. 11, where $\overrightarrow{b} = b_1, \cdots, b_n$.

Finally, there is an operation of *negative substitution* $[t/x]e$, in whose recursive definition the critical equation is $[t/x]x = \mathsf{hd}(t)$.

Polarized normalization. The normalization rules are given in Fig.12. These rules are relations on expressions, with the exception of the rule for $\wedge$, which is a rule on heads, generating a relation on expressions through the property: $H \to H' \Rightarrow [H]\mathcal{S} \to [H']\mathcal{S}$. Then, by compatible closure, one-step reduction relations, denoted $\to$, are generated.

In the first four rules, a *negative redex* has the form $[\mathsf{hd}(t)]\mathcal{S}'$, the last three rules reduce *positive redexes*, which are positive case-expressions. So, if an expression contains a redex, it is not normal. On the other hand, if a typed expression is not normal, then either it contains a positive case-expression (which has to have the form of a positive redex, as the three possible forms of a value are covered by the last three reduction rules), or it contains a negative redex (because the four possible forms of $\mathsf{hd}(t)$ are covered by the first four reduction rules, and typing forces $\mathsf{hd}(t)$ to occur as shown). So, a typed expression is normal iff it is irreducible. Moreover, each reduction rule eliminates a maximal formula (the type of t in $\mathsf{hd}(t)$ or the type of V in $\mathsf{case\,val}(V, \overrightarrow{b})$). So, each reduction rule is a detour conversion rule.

The rule for $\wedge$ is familiar and the easiest. The rule for $\supset$ reduces an ordinary β-redex: a positive case-expression is created, where the first component is V, to be analyzed by the last three reduction rules. These either chose a component of a co-pair $\overrightarrow{b}$ indicated by an injection, or substitute a thunked term, or rename a positive variable. The remaining two rules deal with the situation where the head term is a co-binding and their common idea is that (case)ap and hd cancel out.

4 Results

Isomorphism. Perhaps surprisingly, we prove that $\lambda^{\pm}_{\mathsf{G}}$and $\lambda^{\pm}_{\mathsf{N}}$ are two views of the same system. We profit from a technique that has matured for much simpler logics [5,6].

Obviously one has to adjust the treatment of disjunction in $\lambda^{\pm}_{\mathsf{G}}$ to match that of $\lambda^{\pm}_{\mathsf{N}}$: (i) One adopts an n-ary injection $\mathsf{in}^n_k(V)$; (ii) The right-introduction rule is the same as the introduction rule in $\lambda^{\pm}_{\mathsf{N}}$; (iii) The left-introduction rule becomes

$$\frac{(\Gamma|b_i : B_i \Longrightarrow A)_{i=1,\cdots,n}}{\Gamma|[b_1,\cdots,b_n] : B_1 \vee \cdots \vee B_n \Longrightarrow A}\ \vee_L/\cdot$$

(iv) The reduction rule is now $\langle \mathsf{in}^n_k(V)|[b_1,\cdots,b_n]\rangle \to \langle V|b_k\rangle$.

The key tools are maps $\Theta : \lambda^{\pm}_{\mathsf{G}} \longrightarrow \lambda^{\pm}_{\mathsf{N}}$ and $\Psi : \lambda^{\pm}_{\mathsf{N}} \longrightarrow \lambda^{\pm}_{\mathsf{G}}$:

- For $T = V, t, e \in \lambda^{\pm}_{\mathsf{G}}$, we define the value (resp. term, expression) $\Theta(T)$; for $S \in \lambda^{\pm}_{\mathsf{G}}$ we define the expression $\Theta(H, S)$, given H. The definition is a simultaneous recursion on T and S.
- For $T = V, t, e \in \lambda^{\pm}_{\mathsf{N}}$, we define the value (resp. term, expression) $\Psi(T)$; for $H \in \lambda^{\pm}_{\mathsf{N}}$ we define the expression $\Psi(H, S)$, given S. The definition is a simultaneous recursion on T and H.

Given that there are no co-terms in $\lambda^{\pm}_{\mathsf{N}}$, the translation of co-terms is done "on the fly": basic co-terms abort, $z.e$, $x.e$ are translated homomorphically, while $\Theta([b_1,\cdots,b_n]) = (\Theta(b_1),\cdots,\Theta(b_n))$ and $\Psi(b_1,\cdots,b_n) = [\Psi(b_1),\cdots,\Psi(b_n)]$.

Values and terms are translated homomorphically; so are expressions which are returned values or delivered terms. The remaining clauses for expressions are:

$$\begin{array}{rclrcl}
\Theta(\mathsf{coret}(x,S)) &=& \Theta(x,S) & \Psi(\mathsf{ap}(H)) &=& \Psi(H,\mathsf{nil})\\
\Theta(\langle t|S\rangle) &=& \Theta(\mathsf{hd}(\Theta t),S) & \Psi(\mathsf{case\,ap}(H,\overrightarrow{b})) &=& \Psi(H,\mathsf{cothunk}(\Psi(\overrightarrow{b})))\\
\Theta(\langle V|p\rangle) &=& \mathsf{case\,val}(\Theta V,\Theta p) & \Psi(\mathsf{case\,val}(V,\overrightarrow{b})) &=& \langle \Psi V|\Psi(\overrightarrow{b})\rangle
\end{array}$$

The motor of the translations is the interplay between spines and heads. Given the parameter H (resp. S), $\Theta(H, S)$ (resp. $\Psi(H, S)$) is defined by recursion on S (resp. H) as follows:

$$\begin{array}{rclrcl}
\Theta(H,\mathsf{nil}) &=& \mathsf{ap}(H) & \Psi(x,S) &=& \mathsf{coret}(x,S)\\
\Theta(H,\mathsf{cothunk}(p)) &=& \mathsf{case\,ap}(H,\Theta p) & \Psi(\mathsf{hd}(t),S) &=& \langle \Psi t|S\rangle\\
\Theta(H,V :: S) &=& \Theta(H\Theta V,S) & \Psi(HV,S) &=& \Psi(H,\Psi V :: S)\\
\Theta(H,\mathsf{i} :: S) &=& \Theta(H\mathsf{i},S) & \Psi(H\mathsf{i},S) &=& \Psi(H,\mathsf{i} :: S)
\end{array}$$

For $T = V, t, e$, maps Θ and Ψ preserve types. For H and S, typing is given in Fig.13.

$$\frac{\Gamma \rhd H : N \quad \Gamma[S : N] \vdash A}{\Gamma \vdash \Theta(H, S) : A} \qquad \frac{\Gamma \rhd H : N \quad \Gamma[S : N] \vdash A}{\Gamma \vdash \Psi(H, S) : A}$$

Fig. 13. Typing for Θ and Ψ

Theorem 4.1 (Bijection) *Θ and Ψ are mutual inverses at the levels of values, terms, and expressions, that is: (i) $\Theta\Psi(T) = T$ with $T = V, t, e$ in $\lambda^{\pm}_{\mathsf{N}}$; and (ii) $\Psi\Theta(T') = T'$, with $T' = V, t, e$ in $\lambda^{\pm}_{\mathsf{G}}$.*

Proof. The statement (i) is proved together with (iii) $\Theta\Psi(H, S) = \Theta(H, S)$ for all S; the proof is by simultaneous induction of T and H in $\lambda^{\pm}_{\mathsf{N}}$. The statement (ii) is proved together with (iv) $\Psi\Theta(H, S) = \Psi(H, S)$ for all H; the proof is by simultaneous induction on T' and S in $\lambda^{\pm}_{\mathsf{G}}$. □

The completeness of $\lambda^{\pm}_{\mathsf{N}}$ follows: every intuitionistic theorem has a proof in the system (after being polarized to become a polarized formula) - insofar the same holds of $\lambda^{\pm}_{\mathsf{G}}$ (completeness of focusing [13]) and thanks to the previous theorem.

In order to prove the isomorphism theorem below (Theorem 4.3), we need some preliminary work. We extend Θ and Ψ with maps of spines to spines:

$$\begin{array}{rclrcl}
\Theta(\mathsf{nil}) &=& \mathsf{ap}([\_]) & \Psi(\mathsf{ap}([\_])) &=& \mathsf{nil} \\
\Theta(\mathsf{cothunk}(p)) &=& \mathsf{case\,ap}([\_], \Theta p) & \Psi(\mathsf{case\,ap}([\_], \overrightarrow{b})) &=& \mathsf{cothunk}(\Psi(\overrightarrow{b})) \\
\Theta(V :: S) &=& [[\_]\Theta V]\Theta S & \Psi([[\_]V]\mathcal{S}) &=& \Psi V :: \Psi\mathcal{S} \\
\Theta(\mathsf{i} :: S) &=& [[\_]\mathsf{i}]\Theta S & \Psi([[\_]\mathsf{i}]\mathcal{S}) &=& \mathsf{i} :: \Psi\mathcal{S}
\end{array}$$

It is an immediate consequence of Theorem 4.1 that $\Psi\Theta S = S$ and $\Theta\Psi\mathcal{S} = \mathcal{S}$.

Having extended Θ to spines, it is very handy to express $\Theta(H, S)$ in terms of hole filling. Let $\mathcal{S} = \Theta S$. An easy induction on S establishes $\Theta(H, S) = [H]\mathcal{S}$, from which the equality $\Psi(H, S) = \Psi([H]\mathcal{S})$ follows.

Next we see how Θ maps the various operations of substitution and hole filling.

Lemma 4.2

(i) $\Theta([t/x]T) = [\Theta t/x]\Theta T$, *for* $T = e, V, t', b, p$.

(ii) $\Theta(e[\star\backslash p]) = [\![\Theta e]\!]\Theta p$.

(iii) $\Theta(T@S) = (\Theta T)@(\Theta S)$, *for* $T = b, p$.

(iv) $\Theta([z/z']T) = [z/z']\Theta T$, *for* $T = V, t, S, b, p, e$.

Proof. (i) The statement is proved together with $\Theta([\Theta t/x]H, [t/x]S) = [\Theta t/x]\Theta(H, S)$, for all H. The proof is by simultaneous induction on T and S. (ii) The statement is proved together with: $\Theta(H, S[\star\backslash p]) = [\![\Theta(H, S)]\!]\Theta p$, for all H; and $\Theta(p'[\star\backslash p]) = \Theta(p')@'\Theta(p)$; and $\Theta(b[\star\backslash p]) = \Theta(b)@'\Theta(p)$. The proof is by simultaneous induction on e, S, p' and b. (iii) The statement is proved together with $\Theta(e@S) = [\![\Theta e]\!]\Theta S$. The proof is by simultaneous induction on T and e. (iv) By simultaneous induction on T. □

Using Theorem 4.1, it is immediate to obtain, from the preceding lemma, a similar lemma for Ψ. For instance, corresponding to (ii) we get $\Psi([\![e]\!]\overrightarrow{b}) = (\Psi e)[\star\backslash\Psi\overrightarrow{b}]$.

Theorem 4.3 (Isomorphism) *Θ and Ψ are mutually inverse maps between cut-elimination in $\lambda^{\pm}_{\mathsf{G}}$ and normalization in $\lambda^{\pm}_{\mathsf{N}}$, that is: Let $\rho = \supset, \wedge, \uparrow, a^-, \vee, \downarrow, a^+$ and let $\to_\rho$ denote either the reduction relation in $\lambda^{\pm}_{\mathsf{G}}$ generated by rule ρ in Fig.6, or the reduction relation in $\lambda^{\pm}_{\mathsf{N}}$ generated by rule ρ in Fig.12. Then, for T, T' either values, terms, or expressions:*

(i) *$T \to_\rho T'$ in $\lambda^{\pm}_{\mathsf{G}}$ iff $\Theta(T) \to_\rho \Theta(T')$ in $\lambda^{\pm}_{\mathsf{N}}$.*
(ii) *$T \to_\rho T'$ in $\lambda^{\pm}_{\mathsf{N}}$ iff $\Psi(T) \to_\rho \Psi(T')$ in $\lambda^{\pm}_{\mathsf{G}}$.*

Proof. The "if" statements follow from the "only if" statements and Theorem 4.1. Each "only if" statement is proved by induction on $T \to_\rho T'$, using Lemma 4.2 for Θ and the similar lemma for Ψ. □

Negative identity and η-expansion. In the ordinary λ-calculus, a context $\mathcal{T} = \lambda x.[\_]x$ might be called a η-expander for type $A \supset B$: given M of this type, $\mathcal{T}[M]$ is the η-expansion of M, which is again a term of type $A \supset B$. We analyze this question in $\lambda^{\pm}_{\mathsf{N}}$ because it is related to the question of *negative identity*: we know $\Gamma, x : a^- \vdash \mathsf{ap}(x) : a^-$, but is there an expression $e_N(x)$ (possibly depending on the negative type N) such that $\Gamma, x : N \vdash e_N(x) : N$, for all N? We could call such expression the "generalized negative return", by analogy with the fact $\Gamma, z : a^+ \vdash \mathsf{ret}(z) : a^+$.

Theorem 4.4 (Negative identity) *For all N, there are contexts $\mathcal{T}_N$ and $\mathcal{E}_N$, which are respectively a term and an expression with holes, holes which in both cases expect hole expressions of type N, such that the following rules are admissible:*[5]

$$\frac{\Gamma \rhd H : N}{\Gamma \Longrightarrow \mathcal{T}_N[H] : N} \qquad \frac{\Gamma \rhd H : N}{\Gamma \vdash \mathcal{E}_N[H] : N}$$

Proof. First, for a positive type $P = B_1 \vee \cdots \vee B_n$, for B the k-th operand of this disjunction, we define the macros b_B (a basic co-term) and $\mathcal{B}_{B,N}$ (a basic co-term with 0, 1 or more holes expecting hole expressions of type $P \supset N$)[6]:

$$B = a^+ : b_B = z.\mathsf{ret}(\mathsf{in}^n_k(z)) \qquad \mathcal{B}_{B,N} = z.\mathcal{E}_N[[\_]\mathsf{in}^n_k(z)]$$

$$B = \downarrow M : b_B = x.\mathsf{ret}(\mathsf{in}^n_k(\mathsf{thunk}(\mathcal{T}_M[x]))) \qquad \mathcal{B}_{B,N} = x.\mathcal{E}_N[[\_]\mathsf{in}^n_k(\mathsf{thunk}(\mathcal{T}_M[x]))]$$

$$B = \perp : b_B = \mathsf{abort} \qquad \mathcal{B}_{B,N} = \mathsf{abort}$$

[5] Notice neither $\mathcal{T}_N$ nor $\mathcal{E}_N$ are spines. For these contexts, we resort to the usual hole-filling notation.
[6] In these macros, if $n = 1$, $\mathsf{in}^n_k(V)$ is understood to be V.

$\mathcal{T}_N$ is defined by recursion on N as follows:

$$\begin{array}{ll} N = a^- & : \mathcal{T}_N = \ulcorner \mathsf{ap}([\_]) \urcorner \\ N = \uparrow P, \text{ with } P = B_1 \vee \cdots \vee B_n & : \mathcal{T}_N = \lceil \mathsf{case\,ap}([\_], b_{B_1}, \cdots, b_{B_n}) \rceil \\ N = N_1 \wedge N_2 & : \mathcal{T}_N = \langle \mathcal{T}_{N_1}[[\_]1], \mathcal{T}_{N_2}[[\_]2] \rangle \\ N = P \supset N', \text{ with } P = B_1 \vee \cdots \vee B_n & : \mathcal{T}_N = \lambda(\mathcal{B}_{B_1,N'}, \cdots, \mathcal{B}_{B_n,N'}) \end{array}$$

Finally, for $N = a^-$, we put $\mathcal{E}_N = \mathsf{ap}([\_])$ (which is $\mathcal{T}_N$ with the outer $\ulcorner\cdot\urcorner$ removed); and, for $N = C$, we put $\mathcal{E}_N = \mathsf{dlv}(\mathcal{T}_N)$. The typing of $\mathcal{E}_N[H]$ is immediate from that of $\mathcal{T}_N[H]$. The typing of $\mathcal{T}_N[H]$ is proved by induction on N. The case $N =\uparrow P$ involves proving $\Gamma|b_{B_i} : B_i \longrightarrow P$. The case $N = P \supset N'$ involves proving $\Gamma|\mathcal{B}_{B_i,N'}[H] : B_i \longrightarrow N'$. □

Given a negative type N: $\mathcal{T}_N$ and $\mathcal{E}_N$ are the *η-expanders* of N; given $t : N$, the *η-expansion* of t is $\mathcal{T}_N[\mathsf{hd}(t)]$; $\mathcal{E}_N[x]$ is the *generalized negative return* for N, satisfying $\Gamma, x : N \vdash \mathcal{E}_N[x] : N$. We also obtain the elimination rule for $\downarrow$:

$$\frac{\Gamma \vdash [V :\downarrow N]}{\Gamma \vdash \mathsf{case\,val}(V, x.\mathcal{E}_N[x]) : N} \downarrow_E$$

In terms of programming, as we will soon see, this is a destructor of $\mathsf{thunk}(t)$.

Call-by-push-value. In order to show the expressive power of $\lambda^{\pm}_{\mathsf{N}}$, we are going to simplify the system. Reading positive (resp. negative) types as value types (resp. computation types), and reading $\uparrow$ and $\downarrow$ as the type formers F and U, a variant of CBPV is obtained through the collapse of terms and expressions into a single class of *computations* E, annotating a form of sequents $\Gamma \Vdash E : A$ that replaces the forms $\Gamma \Longrightarrow t : N$ and $\Gamma \vdash e : A$. A right type R is now either a^- or $\uparrow P$. The markers $\mathsf{dlv}(\cdot)$, $\ulcorner\cdot\urcorner$ and $\lceil\cdot\rceil$, which allowed to move between terms and expressions, no longer make sense, but the latter has to be merged with $\mathsf{ret}(V)$ to become $\mathsf{return}(V)$, corresponding to the introduction rule for $\uparrow$. In the definition of $[\![e]\!]\mathcal{S}$, the clause for $\mathsf{dlv}(t)$ has to be replaced by $[\![E]\!]\mathcal{S} = [\mathsf{hd}(E)]\mathcal{S}$, for E a λ-abstraction or a pair. The markers $\ulcorner\cdot\urcorner$ and $\lceil\cdot\rceil$ are erased from reduction rules. The η-expanders $\mathcal{T}_N$ and $\mathcal{E}_N$ collapse to a single one, denoted $\mathcal{E}_N$.

In the resulting system (call it $\lambda^{\pm}_{\mathsf{CBPV}}$) one easily defines variants of CBPV constructors:

$$\begin{array}{ll} \mathsf{let}\, z\, \mathsf{be}\, z'.E := \mathsf{case\,val}(z, z'.E) & E\, \mathsf{to}\, \overrightarrow{b} := \mathsf{case\,ap}(\mathsf{hd}(E), \overrightarrow{b}) \\ \mathsf{pm}\, V\, \mathsf{as}\, \overrightarrow{b} := \mathsf{case\,val}(V, \overrightarrow{b}) & \mathsf{force}(V) := \mathsf{case\,val}(V, x.\mathcal{E}_N[x]) \quad (V :\downarrow N) \end{array}$$

Variants of CBPV's reduction rules become derivable. For instance:

$$\mathsf{return}(V)\, \mathsf{to}\, \overrightarrow{b} \to \mathsf{case\,val}(V, \overrightarrow{b}) \qquad \mathsf{force}(\mathsf{thunk}(E)) \to \mathcal{E}_N[\mathsf{hd}(E)]$$

The first of these rule executes positive return, extracting the value V of positive type that the positive return $\mathsf{return}(\cdot)$ carries, while the second executes negative return, giving to the negative return $\mathcal{E}_N[\cdot]$ the computation E of negative type N it expects; in addition, we see that in $\lambda^{\pm}_{\mathsf{CBPV}}$ forcing causes η-expansion.

5 Discussion

Let us summarize our contribution. We proposed a highly-disciplined system of natural deduction for polarized intuitionistic logic, with several noteworthy features: a proof-search procedure equivalent to focusing; polarized atomic formulas with logical structure: introduction, elimination, normalization rules; elimination rules determined by polarity: a unique, general elimination rule for positive connectives, and dedicated, Gentzen-style, elimination rules for the negative connectives; all normalization rules are detour-conversion rules (no commuting conversions); a perfect equivalence with focused sequent calculus, which gives the ultimate justification for the system's design. In addition, in the Curry-Howard spirit, the proof-system comes with a programming formalism, the polarized λ-calculus $\lambda^{\pm}_{\mathsf{N}}$, which is the key to understand the connection between focusing and CBPV, precisely because it is isomorphic to the focused sequent calculus $\lambda^{\pm}_{\mathsf{G}}$, but is a natural deduction syntax like CBPV.

We regard the recapitulation of focusing in Section 2 as an improvement over Simmons' system [13] in some particular aspects. First, in the design of $\lambda^{\pm}_{\mathsf{G}}$ we were very careful to keep alive all the syntactic distinctions and dualities where they exist. That is why $\lambda^{\pm}_{\mathsf{G}}$ has five syntactic classes and corresponding judgments, with Simmon's class of terms being here decomposed into terms, co-terms and expressions. Such move is within the spirit of focusing and polarization, which aims at a maximally disciplined syntax, and pays off: $\lambda^{\pm}_{\mathsf{G}}$ makes evident the dualities term/co-term, return/co-return, thunk/co-thunk, binding/co-binding, negative substitution/positive substitution. Second, we distinguished, for each inference rule, its inferential (top-down) role from its role in bottom-up proof-search. As a result of this careful design, we see that proof-search comprises, not only the main phases of focusing and inversion, separated by blurring, but also the actions of accepting and collecting with which the stage of inversion ends; and, on the inferential side, we discover that polarized atoms have logical structure, having their own introduction rules.

Even if we do not insist on the logic we treat here, it is difficult to find in the literature works covering the programme we set up in the third paragraph of Section 1. For instance, the natural deduction side of the story is absent in [15,13]; and, in [3], even if a natural deduction system for polarized intuitionistic linear logic is proposed, and connected to a focused sequent calculus from [4], the former was not designed to be the perfect reflection of the latter, nor is any proof-term language developed (the same can be said about the system for classical propositional logic in [7]).

Only in [2] a similarly comprehensive programme is found, and with the goal of studying the connection between polarized intuitionistic logic and CBPV. In *op. cit* a natural deduction system is obtained as the type system of CBPV in bi-directional style; and a variant of the focused sequent calculus LJF [10] together with its λ-calculus is developed. We remark, using our notation, some of the many features of the latter that distinguish it from our $\lambda^{\pm}_{\mathsf{G}}$: a variable may be declared in a context Γ with a type of any polarity; spines annotate sequents $\Gamma[S : N] \vdash M$, with a negative r.h.s. type; the reduction rule for $\supset$ triggers an operation $[V/x]t$, which strikingly contrasts both with our reduction rule and our concepts of positive and negative substitution. The conclusion is drawn that LJF and CBPV "are the same language, up to the question of η-expansion". The present paper may be seen as leading to a similar conclusion; but while *op. cit* gets there after a comparison of abstract machines that were previously associated with each calculus, we go through a careful proof-theoretical analysis that delivers the novel and important polarized λ-calculus.

References

[1] Andreoli, J., *Logic programming with focusing proofs in linear logic*, Journal of Logic and Compututation **2** (1992), pp. 297–347.

[2] Brock-Nannestad, T., N. Guenot and D. Gustafsson, *Computation in focused intuitionistic logic*, in: *Proceedings of PPDP'15* (2015), pp. 43–54.

[3] Brock-Nannestad, T. and C. Schürmann, *Focused natural deduction*, in: *Proceedings of LPAR'10*, Lecture Notes in Computer Science **6397** (2010), pp. 157–171.

[4] Chaudhuri, K., F. Pfenning and G. Price, *A logical characterization of forward and backward chaining in the inverse method*, J. Autom. Reasoning **40** (2008), pp. 133–177.

[5] Espírito Santo, J., *The λ-calculus and the unity of structural proof theory*, Theory of Computing Systems **45** (2009), pp. 963–994.

[6] Espírito Santo, J., *Towards a canonical classical natural deduction system*, Ann. Pure Appl. Logic **164** (2013), pp. 618–650.

[7] Ferrari, M. and C. Fiorentini, *Proof-search in natural deduction calculus for classical propositional logic*, in: H. de Nivelle, editor, *Proceedings TABLEAUX 2015*, Lecture Notes in Computer Science **9323** (2015), pp. 237–252.

[8] Girard, J.-Y., Y. Lafont and P. Taylor, "Proofs and Types," Cambridge University Press, 1989.

[9] Levy, P. B., *Call-by-push-value: Decomposing call-by-value and call-by-name*, Higher-Order and Symbolic Computation **19** (2006), pp. 377–414.

[10] Liang, C. and D. Miller, *Focusing and polarization in linear, intuitionistic, and classical logic*, Theoretical Computer Science **410** (2009), pp. 4747–4768.

[11] Prawitz, D., "Natural Deduction. A Proof-Theoretical Study," Almquist and Wiksell, Stockholm, 1965.

[12] Sieg, W. and S. Cittadini, *Normal natural deduction proofs (in non-classical logics)*, in: *Mechanizing Mathematical Reasoning, Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, Lecture Notes in Computer Science **2605** (2005), pp. 169–191.

[13] Simmons, R. J., *Structural focalization*, ACM Trans. Comput. Log. **15** (2014), pp. 21:1–21:33.

[14] von Plato, J., *Natural deduction with general elimination rules*, Annals of Mathematical Logic **40** (2001), pp. 541–567.

[15] Zeilberger, N., *On the unity of duality*, Ann. Pure Appl. Logic **153** (2008), pp. 66–96.