Call-by-value isn't dual to call-by-name, call-by-name is dual to call-by-value!

Harry G. Mairson*
Computer Science Department
Brandeis University
Waltham, MA 02454, USA
mairson@cs.brandeis.edu

ABSTRACT

Gentzen's sequent calculus for classical logic shows great symmetry: for example, the rule introducing $\wedge$ on the left of a sequent is mirror symmetric to the introduction rule for the dual operator $\vee$ on the right of a sequent. A consequence of this casual observation is that when $\Gamma \vdash \Delta$ is a theorem over operators $\{\vee, \wedge, \neg\}$, then so is $\Delta^\circ \vdash \Gamma^\circ$, where Σ° reverses the order of formulas in Σ, and exchanges each instance of $A \wedge B$ ($C \vee D$) in a formula of Σ with $B \vee A$ ($D \wedge C$). This symmetry of rules means that the duality of *theorems* is also a duality of *proof structures.*

When introducing implication ($\rightarrow$), the duality of theorems can be recovered by a translation (say, $A \rightarrow B \equiv \neg A \vee B$), but the duality of proofs is naively lost. By instead *adding* a logically redundant *coimplication* $\leftarrow$, that duality is restored. A classical Curry-Howard correspondence then dualizes application and λ-abstraction over expression variables, which implement $\rightarrow$, with co-application and co-λ-abstraction over continuation variables, which implement $\leftarrow$. The proof syntax and programming language we recover is Filinski's *symmetric λ-calculus* [3], augmented with a certain facility in naming continuations found in $\lambda\mu$-calculus [17].

The *dual calculus* of [18] lacks λ, encoding it instead using primitive operators—in particular, using computationally redundant *sharing.* It is from such nonlinearity that confluence is lost in normalization of classical proofs, which is why call-by-name and call-by-value translations are used to restore it. Interpreted from a linear logic perspective, this encoding unnecessarily requires *two* copies of every procedure calling context, because the dual calculus lacks the computational power of $\parr$. Further, the call-by-value and call-by-name encodings of λ and @ are not isomorphic, nor do they have duals. The symmetric λ-calculus invites no such complication or redundancy, and its call-by-name and call-by-value CPS translations are dual.

Using a *direct-style translation* that we introduced in [14], where the administrative redexes of CPS translation are eliminated and replaced by a *boxing strategy*, dual CBN and CBV *proofnets* are recovered. These proofnets elaborate the dual, confluent proof structures with explicit sharing, and identify explicitly the coroutined alternation in evaluation of expressions and continuations. The language is then built from four kinds of pairing (with complementary unpairing): of two expressions (a product), two continuations (a coproduct), a continuation and expression (application context), and an expression and continuation (co-application context).

*Supported by NSF Grants CDA-9806718, CCR-0228951, and the Tyson Foundation.

CW04, January 17, Venice, Italy.

1. INTRODUCTION

Ne dites pas 'disez'—disez 'dites.'
–French proverb

Good and bad, sadist and masochist, Heaven and Hell, mind and body, wave and particle, positive and negative, liberal and conservative, optimist and pessimist, female and male, bug and feature, McDonald's and the Tour d'Argent: notions of symmetry and duality are ubiquitous. This sentiment was best put into song by Herman Hupfeld in 1931 [10]:

> *Moonlight and love songs, never out of date*
> *Hearts full of passion, jealousy and hate*
> *Woman needs man, and man must have his mate*
> *That no one can deny.*

No one can deny either that symmetry and duality are also pervasive in logic, for example in the symmetry of rules for proof formation in Gentzen's sequent calculus, and in the consequent duality of conjunction and disjunction. This being the case, we should add to the above list *hypothesis and conclusion*, and through their interpretation via the Curry-Howard correspondence, *expression and continuation.*

Let $\Gamma \vdash \Delta$ be a theorem in sequent calculus, where Γ, Δ are sets of propositional formulas over the operators $\{\vee, \wedge, \neg\}$, with the naive truth-table interpretation that $(\bigwedge \Gamma) \supset (\bigvee \Delta)$; then $\Delta^\circ \vdash \Gamma^\circ$ is also a provable sequent, where Σ° reverses the order of formulas in sequence Σ, and also exchanges each instance of $A \wedge B$ ($C \vee D$) in a formula of Σ with $B \vee A$ ($D \wedge C$).

This symmetry of rules means that the duality of *theorems* is also a duality of *proof structures:* not only is each provable sequent in a one-to-one correspondence with its dual, but through a mirror reflection of the proof trees, the proof structures are in a similar correspondence.

The simple reason is that introduction of $\neg$ on the left and right are dual, just as introduction of $\vee$ ($\wedge$) on the left (right) is dual with introduction of $\wedge$ ($\vee$) on the right (left); for example,

$$\frac{\Delta_1, A \vdash \Gamma_1 \quad \Delta_2, B \vdash \Gamma_2}{\Delta_1, \Delta_2, A \vee B \vdash \Gamma_1, \Gamma_2} (\vee\text{L})$$

is dual with

$$\frac{\Gamma_2 \vdash B, \Delta_2 \quad \Gamma_1 \vdash A, \Delta_1}{\Gamma_2, \Gamma_1 \vdash B \wedge A, \Delta_2, \Delta_1} (\wedge\text{R})$$

The proof of the duality theorem for $\{ \vee, \wedge, \neg \}$ is then easy. Write your proof on a clear transparency sheet, using only symmetric letters (A,H,I,M,O,T,U,V,W,X,Y) for propositional variables, $\boxplus$ for $\vdash$, $\overline{\alpha}$ for $\neg\alpha$, and $\lessdot, \gtrdot$ for $\vee$ and $\wedge$. Then turn the transparency sheet over.

Understanding the duality between call-by-name (CBN) and call-by-value (CBV) evaluation has been the subject of many papers. Because so much of this investigation is linked with understanding the computational content of cut elimination in classical logic, enormous credit must be given to Tim Griffin [9], who first investigated the subject in a way that was accessible and exciting to computer scientists. He saw that the double-negation embeddings of classical into intuitionistic logic, due to Gödel [6] and independently Kolmogorov [12], could be interpreted via the Curry-Howard correspondence, where $\neg A \equiv A \rightarrow \bot$ describes the type of a continuation, and $A \rightarrow \neg\neg A$ has proof $\lambda x : A.\lambda k : \neg A.kx$, not unlike the CBV CPS translation of a variable. Following this work, Chet Murthy's dissertation was an encyclopedic compendium of the varieties of $\neg\neg$-embeddings, and to which flavor CPS translation they each correspond [16].

Recently Wadler [18] proposed a dual characterization of CBV and CBN, adding insights to many other earlier contributions to this genre. As motivation and inspiration, he mentions Filinski's *symmetric λ-calculus* [3], and the *LKQ/LKT calculi* of Danos, Joinet, and Schellinx based on proofnets and linear logic [2]. But "Filinski's formulation lacks any connection with logic," he wrote, adding that the LKQ/LKT formulation "is in terms of proofnets and linear logic, with a less direct connection to computation."

I disagree. Filinski's calculus was developed in the framework of category theory and particularly the idea of duality; through the Curry-Howard correspondence, the connection to logic is very apparent—see, for example, [13]. (Years ago, I offered Phil Wadler $100 for his out-of-print hardbound copy of [13]—he politely and wisely refused.) In addition, proofnets and their normalization are just the linear logician's way of talking about *graph reduction*, a familiar technique from the toolbox of functional programming language implementation—see, for example, [11]. Furthermore, linear logic is the key to understanding *what gets copied and discarded, and when*, in normalizing a classical proof, and it is that understanding which is embodied in the CPS translation. Proofnets permit us to eliminate all the administrative redexes of CPS, replacing them with boxing strategies [14]. The proofnet formulation provides a very clear picture—a *visual language*—of the *coroutining* between evaluation of expressions and continuations that is implicit in the CBV and CBN translations.

Lastly, I would like to argue that the "crisp" dual characterization of CBV and CBN given in [18] is nonetheless a characterization that can be improved. Wadler's *dual calculus* for representing classical proofs does a clean job of explaining the $\{ \vee, \wedge, \neg \}$ fragment, but is less satisfactory in its rendition of λ-abstraction and application, which capture the constructive content of implication ($\rightarrow$). A duality between CBV and CBN *λ-calculus* ought to—at least—explain the duality of λ.

Here are the lacunae in [18] that we attempt to resolve. First, the representations of λ and apply are not primitive, but are derived from other operators. Second, the CBV and CBN CPS representations of $\lambda x.E$, unlike the standard ones, are not isomorphic modulo the translation of E. Third, the interpretations of λ and apply technically have duals, but not ones that are idiomatic programming constructs—like the dual calculus primitives, λ and apply. Finally, the resultant normalization procedure is redundant in a way that mimics neither CBV nor CBN—even though the *results* of normalization mimic CBV and CBN, the intensional *processes* of normalization do not. Recall that a procedure is always called in a context that pairs an input (the argument) with what to do with the output (the continuation). The suggested coding of λ demands *two identical* such pairs, extracting the argument from one, and the continuation from the other. I am reminded of a black housemate of mine in graduate school who was invited to a dinner for women and minorities in the sciences—she exclaimed to me, "I want two tickets!"

In contrast, we present a modified version of the dual calculus, where λ and apply are primitives, the translations of $\lambda x.E$ are just as in the standard CBV and CBN translations, λ and apply have duals, and the normalization that explains procedure call avoids useless duplication. Duality of λ and apply comes from simply *adding* their duals to the calculus—the insight of Filinski—just as the fundamental theorem of algebra is proved by completing the reals with $\sqrt{-1}$. The redundancy problem is solved by the *usual* linear logic primitive coding of λ as $\invamp$, which works as a linear *unpairing* of a call site.

This improvement results in a dual calculus that is made up of four kinds of pairing: of two expressions (a product), two continuations (a coproduct), a continuation and expression (application context), and an expression and continuation (co-application context). The complementary *unpairings* for product and coproduct are nonlinear (they discard one component), but are linear for application and co-application.

We do not claim our characterization of duality to be *crisp*, but we do insist that it is a *flaky formulation*—which is, certainly if you are thinking about pie crust, a very good thing. It remains an open problem to figure out what is in turn wrong with our solution, and to come up with something better.

2. CLASSICAL CURRY-HOWARD CORRESPONDENCE: A CRUDE FIRST APPROXIMATION

A duality between CBV and CBN is really the solution of two separate problems. The first is a dualization of classical logic at the level of proof as well as at the level of theoremhood, allowing nonconfluence. The second is to elaborate—symmetrically—the former solution so as to maintain confluence. (It is worth noting that classical logic is dually nonconfluent! Any nonconfluence in the normalization of $\Gamma \vdash \Delta$ is dual to a nonconfluent normalization of $\Gamma^\circ \vdash \Delta^\circ$.) The former is best achieved by making primitives of λ and application, comprising the constructive content of $\rightarrow$. But what, then, is the dual of λ? The simple, lovely answer is co-λ, which abstracts over a continuation, producing a continuation. Similarly, co-application is the context which provides a paired expression and continuation to be supplied to a co-λ-abstraction. These co-operations implement the logical operation $\leftarrow$, where $B \leftarrow A$ is truth-table equivalent to $\neg(B \rightarrow A)$; observe that from proofs of $B \leftarrow A \vdash$ and $A \vdash$ we may derive $B \vdash$. Then the duality theorem says $\Gamma \vdash \Delta$ iff $\Delta^\circ \vdash \Gamma^\circ$, where Σ° denotes the reversal of the sequence of formulas given in Σ with $A \vee B$

exchanged with $B \wedge A$, and $A \to B$ exchanged with $B \leftarrow A$.

In Figure 1 we present a classical sequent calculus in the style of Gentzen [4], which we call the *Filinski symmetric sequent calculus* because it includes the connective $\leftarrow$ as in Filinski's type system for symmetric λ-calculus [3]. The calculus includes programs which, according to the Curry-Howard correspondence, describe the proof. We write $E \blacktriangleright \Gamma \vdash \Delta$, where E is a program that describes the structure of the proof of sequent $\Gamma \vdash \Delta$, and where Γ (Δ) is a sequence of propositional formulas giving the type of each input *expression* (output *continuation*). At most one formula in $\Gamma \cup \Delta$ is said to be *principal* and is underlined; all other formulas are annotated with a unique *name*, ranging over Roman (Greek) letters for each formula in Γ (Δ). E is called an *expression* or *term* if it has a principal output (in Δ), a *continuation* or *coterm* if it has a principal input (in Γ), and is otherwise called a *statement*.

In the presentation of the calculus, the dual sequents are mirror images, right up to reversal—just like turning over a transparency.[1] Rules ($\to$ L) and ($\leftarrow$ R) are a good example. Observe that rule (CUT) is self-dual.

To identify further this duality, we specify it by a transformation on the associated proof terms. Assume that $(-)^\circ$ is a bijection between Roman expression variables and Greek continuation variables; abusing notation, we write the inverse function identically. Now we extend this bijection to describe the dual of each program:

$$\begin{array}{rclcrcl}
(\langle M, N\rangle)^\circ & = & [M^\circ, N^\circ] & & ([K, L])^\circ & = & \langle K^\circ, L^\circ\rangle \\
(\langle M\rangle\mathsf{inl})^\circ & = & \mathsf{fst}[M^\circ] & & (\mathsf{fst}[K])^\circ & = & \langle K^\circ\rangle\mathsf{inl} \\
(\langle N\rangle\mathsf{inr})^\circ & = & \mathsf{snd}[N^\circ] & & (\mathsf{snd}[K])^\circ & = & \langle K^\circ\rangle\mathsf{inr} \\
([K]\mathsf{not})^\circ & = & \mathsf{not}\langle K^\circ\rangle & & (\mathsf{not}\langle M\rangle)^\circ & = & [M^\circ]\mathsf{not} \\
((S).\alpha)^\circ & = & \alpha^\circ.(S^\circ) & & (x.(S))^\circ & = & ((S^\circ).x^\circ) \\
(\lambda x.M)^\circ & = & \lambda^\circ x^\circ.M^\circ & & (\lambda^\circ\alpha.K)^\circ & = & \lambda\alpha^\circ.K^\circ \\
(K@^\circ M)^\circ & = & K^\circ @ M^\circ & & (M@K)^\circ & = & M^\circ @^\circ K^\circ
\end{array}$$

$$(M \bullet K)^\circ = K^\circ \bullet M^\circ$$

In the above, M, N range over expressions, K, L over continuations, and S over statements. The left (right) column lists expressions (continuations), and $M \bullet K$ is a statement.

Proposition 2.1. *$(-)^\circ$ is an isomorphism between expressions and continuations, an endomorphism on statements, and is an involution: $P^{\circ\circ} = P$.* □

The Curry-Howard correspondence lets us describe cut elimination by a reduction on the associated proof terms, for example:

$$\begin{array}{lrcl}
(\wedge L) & \langle M, N\rangle \bullet \mathsf{fst}[K] & \longrightarrow & M \bullet K \\
(\wedge R) & \langle M, N\rangle \bullet \mathsf{snd}[K] & \longrightarrow & N \bullet K \\
(\vee L) & \langle M\rangle\mathsf{inl} \bullet [K, L] & \longrightarrow & M \bullet K \\
(\vee R) & \langle M\rangle\mathsf{inr} \bullet [K, L] & \longrightarrow & M \bullet L \\
(\neg) & [K]\mathsf{not} \bullet \mathsf{not}\langle M\rangle & \longrightarrow & M \bullet K \\
(\to) & \lambda x.N \bullet M@K & \longrightarrow & N\{\, M/x\,\} \bullet K \\
(\leftarrow) & K@^\circ M \bullet \lambda^\circ\alpha.L & \longrightarrow & M \bullet L\{\, K/\alpha\,\} \\
(.L) & (S).\alpha \bullet K & \longrightarrow & S\{\, K/\alpha\,\} \\
(.R) & M \bullet x.(S) & \longrightarrow & S\{\, M/x\,\}
\end{array}$$

This calculus is pretty, but not confluent. The example in [3, 18] is perhaps the simplest one: $(x \bullet \alpha).\beta \bullet y.(z \bullet \gamma)$ reduces to either $x \bullet \alpha$ or $z \bullet \gamma$, and follows the standard anomaly in classical deduction (see, for example, [5]):

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash \Delta}}{\Gamma \vdash \Delta, A}\,(\text{W-R}) \qquad \dfrac{\dfrac{\vdots}{\Sigma \vdash \Lambda}}{A, \Sigma \vdash \Lambda}\,(\text{W-L})}{\Gamma, \Sigma \vdash \Delta, \Lambda}\,(\text{CUT})$$

which normalizes via repeated weakenings to either of

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash \Delta}}{\Gamma, \Sigma \vdash \Delta, \Lambda} \qquad \dfrac{\dfrac{\vdots}{\Sigma \vdash \Lambda}}{\Gamma, \Sigma \vdash \Delta, \Lambda}$$

Nonconfluence of classical logic is a standard rationale for linear logic, but in the original state of linearity, confluence is a given. As a consequence, consider a proof $T \blacktriangleright \Gamma \vdash \Delta$ where any variable occurs in T exactly once. Then since the normalization of T (and T°) is confluent, we conclude:

Proposition 2.2. *The linear λ-calculus is self-dual.* □

Similarly,

Proposition 2.3. *The λI-calculus is self-dual.* □

Furthermore, recognize that duality exists even in the face of nonconfluence:

Proposition 2.4. *If $T \blacktriangleright \Gamma \vdash \Delta$ and T reduces to T', then we also have $T^\circ \blacktriangleright \Delta^\circ \vdash \Gamma^\circ$, where T° reduces to T'°.* □

3. CBV AND CBN CPS TRANSLATIONS: CONFLUENCE RESTORED

To restore confluence, we define *values* V, W and *covalues* P, Q inductively as:

$$\begin{array}{rcl}
V, W & ::= & x \mid \langle V, W\rangle \mid \langle V\rangle\mathsf{inl} \mid \langle V\rangle\mathsf{inr} \mid [K]\mathsf{not} \mid \lambda x.N \mid K@^\circ V \\
P, Q & ::= & \alpha \mid [P, Q] \mid \mathsf{fst}[P] \mid \mathsf{snd}[P] \mid \mathsf{not}\langle M\rangle \mid M@Q \mid \lambda^\circ\alpha.K
\end{array}$$

When M, N (K, L) are values (covalues), then a reduction above is *call-by-value* (*call-by-name*).

Call-by-value CPS translation

$$\begin{array}{rcl}
x^v & = & \lambda\gamma.\gamma x \\
(\lambda x.M)^v & = & \lambda\gamma.\gamma(\lambda x.\lambda\kappa.M^v\kappa) \\
(K@^\circ M)^v & = & \lambda\gamma.\gamma(\lambda z.M^v(zK^v)) \\
(\langle M, N\rangle)^v & = & \lambda\gamma.M^v(\lambda m.N^v(\lambda n.\gamma\langle m, n\rangle)) \\
(\langle M\rangle\mathsf{inr})^v & = & \lambda\gamma.M^v(\lambda m.\gamma(\mathsf{inr}\, m)) \\
(\langle M\rangle\mathsf{inl})^v & = & \lambda\gamma.M^v(\lambda m.\gamma(\mathsf{inl}\, m)) \\
([K]\mathsf{not})^v & = & \lambda\gamma.\gamma(\lambda z.K^v z) =_\eta \lambda\gamma.\gamma K^v \\
((S).\alpha)^v & = & \lambda\alpha.S^v
\end{array}$$

[1] We resolve a minor asymmetry in [18], so that both ($\wedge$R) and ($\vee$L) merge contexts; there the former is treated additively, and the latter multiplicatively.

$$
\begin{array}{rcl}
\alpha^v & = & \lambda z.\alpha z \\
(\lambda^\circ \alpha.K)^v & = & \lambda z.z(\lambda\alpha.\lambda x.K^v x) \\
(M@K)^v & = & \lambda z.M^v(\lambda m.zmK^v) \\
([K,L])^v & = & \lambda z.\mathsf{case}\ z\ \mathsf{of}\ \mathsf{inl}\ x \Rightarrow K^v x, \mathsf{inr}\ x \Rightarrow L^v y \\
(\mathsf{fst}[K])^v & = & \lambda z.\mathsf{case}\ z\ \mathsf{of}\ \langle x,-\rangle \Rightarrow K^v x \\
(\mathsf{snd}[L])^v & = & \lambda z.\mathsf{case}\ z\ \mathsf{of}\ \langle -,y\rangle \Rightarrow L^v y \\
(\mathsf{not}\langle M\rangle)^v & = & \lambda z.(\lambda\gamma.M^v\gamma)z =_\eta M^v \\
(x.(S))^v & = & \lambda x.S^v \\
\\
(M \bullet K)^v & = & M^v K^v
\end{array}
$$

Call-by-name CPS translation

$$
\begin{array}{rcl}
x^n & = & \lambda\gamma.x\gamma \\
(\lambda x.M)^n & = & \lambda\gamma.\gamma(\lambda x.\lambda\kappa.M^n\kappa) \\
(K@^\circ M)^n & = & \lambda\gamma.K^n(\lambda\alpha.\gamma\alpha M^n) \\
(\langle M,N\rangle)^n & = & \lambda\gamma.\mathsf{case}\ \gamma\ \mathsf{of}\ \mathsf{inl}\ \alpha \Rightarrow M^n\alpha, \mathsf{inr}\ \beta \Rightarrow N^n\beta \\
(\langle M\rangle\mathsf{inr})^n & = & \lambda\gamma.\mathsf{case}\ \gamma\ \mathsf{of}\ \langle\alpha,-\rangle \Rightarrow M^n\alpha \\
(\langle M\rangle\mathsf{inl})^n & = & \lambda\gamma.\mathsf{case}\ \gamma\ \mathsf{of}\ \langle -,\beta\rangle \Rightarrow N^n\beta \\
([K]\mathsf{not})^n & = & \lambda\gamma.(\lambda z.K^n z)\gamma =_\eta K^n \\
((S).\alpha)^n & = & \lambda\alpha.S^n \\
\\
\alpha^n & = & \lambda z.z\alpha \\
(\lambda^\circ\alpha.K)^n & = & \lambda z.z(\lambda\alpha.\lambda x.K^n x) \\
(M@K)^n & = & \lambda z.z(\lambda\alpha.K^n(\alpha M^n)) \\
([K,L])^n & = & \lambda z.K^n(\lambda\alpha.L^n(\lambda\beta.z\langle\alpha,\beta\rangle)) \\
(\mathsf{fst}[K])^n & = & \lambda z.K^n(\lambda\alpha.z(\mathsf{inl}\ \alpha)) \\
(\mathsf{snd}[L])^n & = & \lambda z.L^n(\lambda\beta.z(\mathsf{inr}\ \beta)) \\
(\mathsf{not}\langle M\rangle)^n & = & \lambda z.z(\lambda\gamma.M^n\gamma) =_\eta \lambda z.zM^n \\
(x.(S))^n & = & \lambda x.S^n \\
\\
(M \bullet K)^n & = & K^n M^n
\end{array}
$$

The novelty in this presentation is just that of co-abstraction and co-application. Since the CBV and CBN CPS translations of λ are identical, so are their duals: we have four versions of the same thing. But where do these crazy translations of @ and $@^\circ$ come from? I know the CBx CPS translations of $\lambda x.P$, $(\lambda x.P)M$—I looked them up. And the equation

$$(\lambda x.P \bullet M@K)^{\aleph} = ((\lambda x.P)M \bullet K)^{\aleph}$$

has to be true for $\aleph \in \{v, n\}$. Furthermore, we know $(N \bullet L)^v = N^v L^v$ and $(N \bullet L)^n = L^n N^n$. As a consequence, we need only solve

$$
\begin{array}{rcl}
(\lambda x.P)^v (M@K)^v & = & ((\lambda x.P)M)^v K^v \\
(M@K)^n (\lambda x.P)^n & = & K^n((\lambda x.P)M)^n
\end{array}
$$

These are nothing but linear (algebra, not logic!) equations in λ-calculus, where the "variables" are the $(M@K)^{\aleph}$. (You can check by substituting that the solutions are correct.) And once we solve for the CBV and CBN @ (the only one where we have pedestrian intuitions!), the two $@^\circ$ come from duality.

4. PROOFNETS: EXCHANGING ADMINISTRATIVE REDEXES FOR A BOXING STRATEGY

We briefly sketch the construction of *proofnet graphs* to implement λ-calculus; more details can be found elsewhere [1, 7, 8]. *Proofnets* are composed of wires and fixed-arity nodes, as well as *boxes*, which enclose subgraphs. The λ-calculus is encoded using apply (@) nodes, λ-nodes, sharing nodes (∇), weakening nodes ($\odot$), and croissants ($\frown$). A box allows a contained subgraph to be duplicated by a sharing node, or discarded by a weakening node. When sharing is no longer required, a croissant can open the box, allowing interaction with the subgraph inside. The meaning of the other nodes should be intuitive.

One port of each node or box is designated as the *principal port*. Other ports are *auxiliary ports*. Reduction takes place when two graph constructs are connected at their principal ports. A box can also interact with another box at its auxiliary port. Global reduction rules are shown in Figure 2(a), where black dots indicate principal ports. Graphs can also be reduced using local rules to simulate manipulation of boxes; these local reductions implement Lévy's *optimal reduction* [15].

In the implementation of λ-calculus within this framework, the big question is where to put the boxes, and how to orient their ports. The solution to this problem determines how subgraphs (representing terms or coterms) can be shared, and how global reductions can be sequentialized. The latter is key in resolving issues of confluence.

There are two standard solutions for the λ-calculus: in linear logic notation, they are usually written $\langle A \to B\rangle \equiv !(\langle A\rangle \multimap \langle B\rangle)$, and $[A \to B] = ![A] \multimap [B]$. The former says: draw a box around each λ-abstraction, and locate a croissant at each *call site* (@) to open the box, so the λ-abstraction can be used. The latter says: draw a box around each *argument* of an application, and locate a croissant at each wire denoting a *variable occurrence*, so that occurrence can open the box and access its contents.

To construct proofnets for the Filinski symmetric calculus, we use the former solution, modified as follows. Given $T \blacktriangleright \Gamma \vdash \Delta$, take the CBx CPS translation of T, and modify its coding as a graph so that wires typed $\bot$ are eliminated, as in Figure 2(b). This *direct-style translation* (see [14]), will for example translate the λ-term $\lambda h : \neg(A \vee \neg A).h(\mathsf{inr}^{\neg A \to A \vee \neg A}(\lambda s : A.\mathsf{inl}.h(\mathsf{inl}^{A \to A \vee \neg A} s)))$ of type $\neg\neg(A \vee \neg A)$ to a proofnet of type $A \vee \neg A$ (though no longer a λ-term).

In Figure 2(c) we show the direct-style translation of the CBV CPS translation of $\langle M, N\rangle \bullet \mathsf{fst}[K]$—notice how the box structure effects a certain synchronization of reductions. Moreover, in Figure 2(d) we show the same translation of $(x \bullet \alpha).\beta \bullet y.(z \bullet \gamma)$; observe how boxing and the weakening node serve to discard half of the computation. Finally, in Figure 2(e) we show the direct-style translation of $(M@K \bullet \lambda x.E)^n$. We will elaborate in greater detail upon the properties of this translation in a subsequent version of this abstract.

Quite apart from issues of boxing, we wish to further emphasize the difference between the coding of λ-abstraction in terms of other primitives, as in [18], with the approach taken above. In the former, the CBV and CBN codings of $\lambda x.E$ are:

$$
\begin{array}{rcl}
(\lambda x.E)_{\mathrm{CBV}} & \equiv & [z.(z \bullet \mathsf{fst}[x.(z \bullet \mathsf{snd}[\mathsf{not}\langle N\rangle])])]\mathsf{not} \\
(\lambda x.E)_{\mathrm{CBN}} & \equiv & (\langle[x.(\langle N\rangle\mathsf{inr} \bullet \gamma)]\mathsf{not}\rangle\mathsf{inl} \bullet \gamma).\gamma
\end{array}
$$

The proofnet codings of these translations appear in Figure 3. Notice that the sharing of z in the CBV coding, and γ in the CBN coding, are represented by sharing nodes in their respective graphs.

The reason is that the selectors used to extract components of the application context (the input expression, and the output continuation) each discard the other component. As a consequence, two copies of the application context are needed. This useless duplication is a consequence of a calculus of proofs that lacks *linear unpairing* (take apart the pair, and use both components). Of course, that is exactly the role of λ when interpreted as $\parr$ in the underlying proof calculus.

5. SUMMARY

It is reasonable to say that duality is very much a consequence of symmetry. In the presentation of the Filinski symmetric λ-calculus, the abstraction over expressions is symmetric to the abstraction over continuations. These notions of λ-symmetry are not derived, but instead primitive in the calculus. This primitive treatment extends the clarity of Wadler's presentation of the $\{ \vee, \wedge, \neg \}$ fragment to implication ($\rightarrow$) and its dual co-implication ($\leftarrow$). I think the clear treatment of these is very important because they are the logical counterpart of λ-abstraction (over expressions as well as continuations) and application—certainly of interest to any functional programmer.

I am still bothered by the situation of the entire development in essentially the multiplicative-exponential part of linear logic—but I would be even more bothered if additives appeared, since the proofnet technology for handling them is far less developed.

This brief abstract has been written quite quickly, and I look forward to expanding the basic direction of its investigation and filling in a variety of gaps. In particular, in an extended form of this abstract I want to address the mechanism of discarding boxes and how to discard certain disconnected components in a box.

I conclude by paraphrasing some observations made by Andrzej Filinski in private communication. Duality may well be a dually-edged sword: it gives insight into similarities, but in the end, what is the point of saying everything twice? It's the department of redundancy department. This investigation really becomes interesting when an object of study and its dual share some, but not all characteristics. Given that call-by-name and call-by-value have dual characteristics, what more can we say about the essential differences between them?

Acknowledgements. Phil Wadler gave his usual spirited and compelling presentation at ICFP 2003 in his characterization of the duality of CBV and CBN, which got me interested in the subject. I've adopted a lot of his notation—though greatly simplifying his Curry-Howard version of classical logic—so that this paper can be read as a sequel to his contribution. Ken Shan started a reading group on continuations at Harvard in autumn 2003, which gave me the opportunity to really understand and analyze Phil's paper. Alan Bawden, Olivier Danvy, and Andrzej Filinski gave me lots of advice and were very useful sounding boards. Finally, I thank Julia Lawall—who I think knows all of this already—for having shown me, years ago, what I don't understand about continuation-passing style.

6. REFERENCES

[1] A. Asperti and S. Guerrini. *The Optimal Implementation of Functional Programming Languages.* CUP, 1998.

[2] V. Danos, J.-B. Joinet, and H. Schellinx. LKQ and LKT: Sequent calculi for second order logic based upon dual linear decompositions of classical implication. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic: Proc. of Linear Logic Wksh., Ithaca, NY, USA, 14–18 June 1993*, volume 222 of *London Math. Society Lecture Notes Series*, pages 211–224. Cambridge University Press, 1995.

[3] A. Filinski. Declarative continuations: An investigation of duality in programming language semantics. In D. H. Pitt, D. E. Rydeheard, P. Dybjer, A. M. Pitts, and A. Poigné, editors, *Proceedings 3th Int. Conf. on Category Theory and Computer Science, CTCS'89, Manchester, UK, 5–8 Sept 1989*, volume 389 of *Lecture Notes in Computer Science*, pages 224–249. Springer-Verlag, Berlin, 1989.

[4] G. Gentzen. *Collected Works. Edited by M.E. Szabo*. North-Holland, Amsterdam, 1969.

[5] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science 7. Cambridge University Press, 1988.

[6] K. Gödel. On intuitionistic arithmetic and number theory (1933). In S. Feferman, editor, *Kurt Gödel: Collected Works*, volume I, pages 287–295. Cambridge University Press, 1986.

[7] G. Gonthier, M. Abadi, and J.-J. Lévy. The geometry of optimal lambda reduction. In *Proc. 19-th Annual ACM Symposium on Principles of Programming Languages, Albuquerque, New Mexico*. ACM Press, New York, NY, January 1992.

[8] G. Gonthier, M. Abadi, and J.-J. Lévy. Linear logic without boxes. In *Proceedings 7th Annual IEEE Symp. on Logic in Computer Science, LICS'92, Santa Cruz, CA, USA, 22–25 June 1992*, pages 223–34. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[9] T. G. Griffin. A formulae-as-type notion of control. In *POPL '90. Proceedings of the seventeenth annual ACM symposium on Principles of programming languages, January 17–19, 1990, San Francisco, CA*, pages 47–58, New York, NY, USA, 1990. ACM Press.

[10] H. Hupfeld. As time goes by (1931). This song is best remembered from the movie *Casablanca*.

[11] S. L. P. Jones. *The Implementation of Functional Programming Languages*. Computer Science. Prentice-Hall, 1987.

[12] A. N. Kolmogorov. On the principle of excluded middle (1925). In J. van Heijenoort, editor, *From Frege to Gödel: A Source Book in Mathematical Logic*, pages 414–437. Harvard University Press, 1967.

[13] J. Lambek and P. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge University Press, Cambridge, 1986.

[14] J. L. Lawall and H. G. Mairson. Sharing continuations: proofnets for languages with explicit control. In *ESOP 2000: 9th European Symposium on Programming, Edinburgh, U.K.*, volume 1782 of *LNCS*, pages 245–259. Springer-Verlag, 2000.

[15] J.-J. Lévy. Optimal reductions in the lambda-calculus. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, pages 159–191. Academic Press, 1980.

[16] C. R. Murthy. Extracting constructive content from classical proofs. Technical Report TR90-1151, Cornell University, Computer Science Department, Aug. 1990.

[17] M. Parigot. Lambda-mu-calculus: an algorithmic interpretation of classical natural deduction. In A. Voronkov, editor, *Logic Programming and Automated Reasoning: International Conference LPAR '92 Proceedings, St. Petersburg, Russia*, pages 190–201, Berlin, DE, 1992. Springer-Verlag.

[18] P. Wadler. Call-by-value is dual to call-by-name. In *Conference record of the Eighth ACM SIGPLAN International Conference on Functional Programming*, pages 189–202, New York, NY, USA, 2003. ACM Press.

Identity group

$$\frac{}{\alpha \blacktriangleright \underline{A} \vdash \alpha : A}\,(\text{Ax-L}) \qquad \frac{}{x \blacktriangleright x : A \vdash \underline{A}}\,(\text{Ax-R}) \qquad \frac{M \blacktriangleright \Gamma \vdash \Delta, \underline{A} \quad K \blacktriangleright \underline{A}, \Sigma \vdash \Lambda}{M \bullet K \blacktriangleright \Gamma, \Sigma \vdash \Delta, \Lambda}\,(\text{Cut})$$

Logical rules

$$\frac{M \blacktriangleright \Gamma \vdash \underline{A}, \Delta \quad N \blacktriangleright \Sigma \vdash \underline{B}, \Lambda}{\langle M, N\rangle \blacktriangleright \Gamma, \Sigma \vdash \underline{A \wedge B}, \Delta, \Lambda}\,(\wedge\text{R}) \qquad \frac{K \blacktriangleright \underline{A}, \Gamma \vdash \Delta}{\mathsf{fst}[K] \blacktriangleright \underline{A \wedge B}, \Gamma \vdash \Delta}\,(\wedge\text{L}_1) \qquad \frac{L \blacktriangleright \underline{B}, \Gamma \vdash \Delta}{\mathsf{snd}[L] \blacktriangleright \underline{A \wedge B}, \Gamma \vdash \Delta}\,(\wedge\text{L}_2)$$

$$\frac{K \blacktriangleright \Lambda, \underline{B} \vdash \Sigma \quad L \blacktriangleright \Delta, \underline{A} \vdash \Gamma}{[K, L] \blacktriangleright \Lambda, \Delta, \underline{B \vee A} \vdash \Sigma, \Gamma}\,(\vee\text{L}) \qquad \frac{M \blacktriangleright \Delta \vdash \Gamma, \underline{A}}{\langle M\rangle\mathsf{inl} \blacktriangleright \Delta \vdash \Gamma, \underline{B \vee A}}\,(\vee\text{R}_1) \qquad \frac{N \blacktriangleright \Delta \vdash \Gamma, \underline{B}}{\langle N\rangle\mathsf{inr} \blacktriangleright \Delta \vdash \Gamma, \underline{B \vee A}}\,(\vee\text{R}_2)$$

$$\frac{K \blacktriangleright \Gamma, \underline{A} \vdash \Delta}{[K]\mathsf{not} \blacktriangleright \Gamma \vdash \underline{\neg A}, \Delta}\,(\neg\text{R}) \qquad \frac{M \blacktriangleright \Delta \vdash \underline{A}, \Gamma}{\mathsf{not}\langle M\rangle \blacktriangleright \Delta, \underline{\neg A} \vdash \Gamma}\,(\neg\text{L})$$

$$\frac{M \blacktriangleright x : A, \Gamma \vdash \underline{B}, \Delta}{\lambda x.M \blacktriangleright \Gamma \vdash \underline{A \to B}, \Delta}\,(\to\text{R}) \qquad \frac{M \blacktriangleright \Gamma \vdash \underline{A}, \Delta \quad K \blacktriangleright \Sigma, \underline{B} \vdash \Lambda}{M@K \blacktriangleright \Gamma, \Sigma, \underline{A \to B} \vdash \Delta, \Lambda}\,(\to\text{L})$$

$$\frac{M \blacktriangleright \Lambda \vdash \underline{B}, \Sigma \quad K \blacktriangleright \Delta, \underline{A} \vdash \Gamma}{K@^{\circ}M \blacktriangleright \Lambda, \Delta \vdash \underline{B \leftarrow A}, \Sigma, \Gamma}\,(\leftarrow\text{R}) \qquad \frac{K \blacktriangleright \Delta, \underline{B} \vdash \Gamma, \alpha : A}{\lambda^{\circ}\alpha.K \blacktriangleright \Delta, \underline{B \leftarrow A} \vdash \Gamma}\,(\leftarrow\text{L})$$

Structural rules

$$\frac{S \blacktriangleright \Gamma, x : A \vdash \Delta}{x.S \blacktriangleright \Gamma, \underline{A} \vdash \Delta}\,(\text{Sel-L}) \qquad \frac{S \blacktriangleright \Delta \vdash \alpha : A, \Gamma}{S.\alpha \blacktriangleright \Delta \vdash \underline{A}, \Gamma}\,(\text{Sel-R})$$

$$\frac{P \blacktriangleright \Gamma \vdash \Delta}{P \blacktriangleright \Gamma, x : A \vdash \Delta}\,(\text{W-L}) \qquad \frac{P \blacktriangleright \Delta \vdash \Gamma}{P \blacktriangleright \Delta \vdash \alpha : A, \Gamma}\,(\text{W-R})$$

$$\frac{P \blacktriangleright \Gamma, x : A, y : A \vdash \Delta}{[x/y]P \blacktriangleright \Gamma, x : A \vdash \Delta}\,(\text{C-L}) \qquad \frac{P \blacktriangleright \Delta \vdash \beta : A, \alpha : A, \Gamma}{[\alpha/\beta]P \blacktriangleright \Delta \vdash \alpha : A, \Gamma}\,(\text{C-R})$$

$$\frac{P \blacktriangleright \Gamma, x : A, y : B, \Sigma \vdash \Delta}{P \blacktriangleright \Gamma, y : B, x : A, \Sigma \vdash \Delta}\,(\text{Int-L}) \qquad \frac{P \blacktriangleright \Delta \vdash \Sigma, \beta : B, \alpha : A, \Gamma}{P \blacktriangleright \Delta \vdash \Sigma, \alpha : A, \beta : B, \Gamma}\,(\text{Int-R})$$

Figure 1: *Filinski's symmetric sequent calculus. Note S, E, K denote statements, expressions, and continuations; P denotes any of the three. An expression always has a principal (underlined) formula to the right of the $\vdash$; a continuation has its principal formula to the left of the $\vdash$.* **Only two rules here are at all new, or at least novel:** *($\leftarrow$ L) which defines co-λ, and ($\leftarrow$ R) which defines co-apply.*

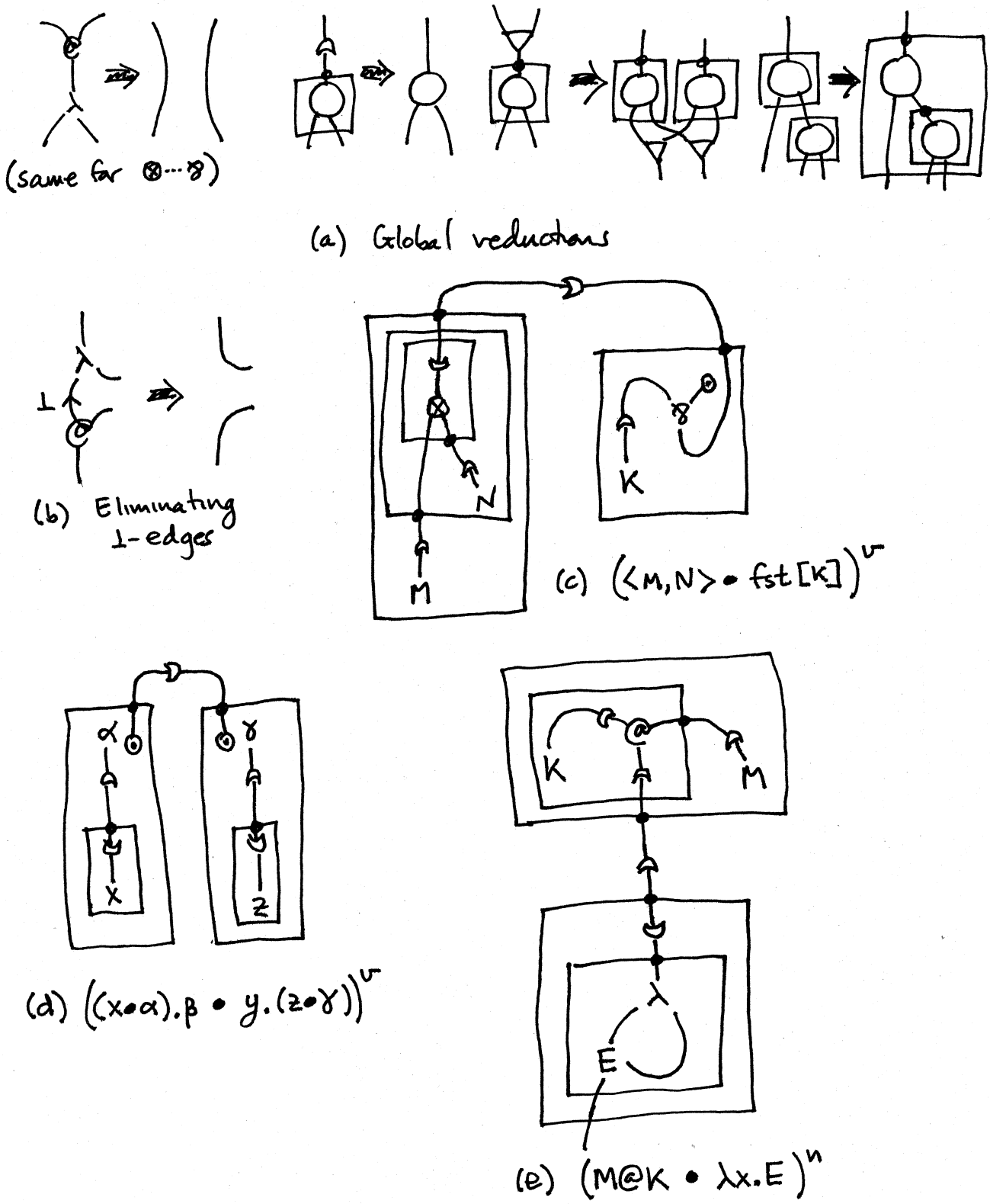

Figure 2: Interpretation of Filinski's symmetric sequent calculus via CPS and direct-style translations.

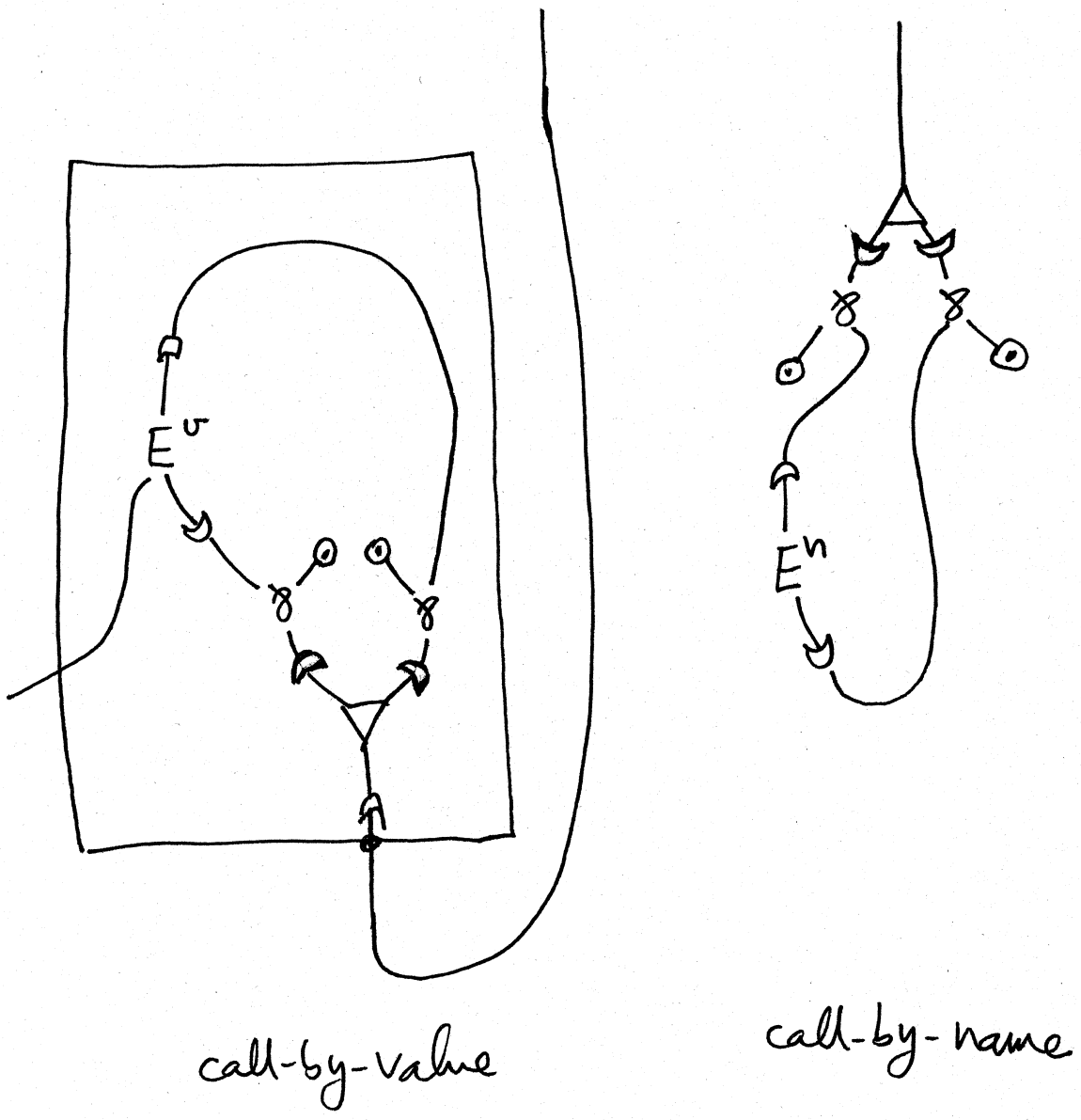

Figure 3: Call-by-value and call-by-name proofnet translations of $\lambda x.E$ in Wadler's dual calculus.