Understanding Call By Push Value through $\overline{\lambda}\mu\tilde{\mu}$

Steffen van Bakel, Emma Tye, and Nicholas Wu

Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2BZ, UK

Abstract

Call by Push Value (CBPV) is a calculus for explicitly modeling call-by-name and call-by-value reduction in the λ-calculus and many other intuitionistic logic-based calculi. We investigate whether classical logic-based calculi, specifically the $\overline{\lambda}\mu\tilde{\mu}$-calculus, can model CBPV and therefore subsume the call-by-name and call-by-value paradigms.
In order to achieve reduction respecting mappings for the CBN and CBV λ-calculus, we define eCBPV, *essential* CBPV, by modifying the syntax slightly by allowing forcing only for variables, thunking only for computations that are not forced variables, and changing the nature of term substitution, also abolishing the U-reduction rule. These changes make it possible to establish a strong relation between $\overline{\lambda}\mu\tilde{\mu}$ and eCBPV, allowing to simulate eCBPV reduction in $\overline{\lambda}\mu\tilde{\mu}$, and preserving assignable types. This should open up avenues for describing the semantics of call-by-name and call-by-value via classical logic semantics.

Introduction

Both the 'call by name' (CBN) and 'call by value' (CBV) reduction strategies of the λ-calculus [4, 2] are important paradigms in theory and as well as in practice of computer science. The first, also known as 'lazy' reduction, is based on the concept that the execution of function application (the contraction of a β-redex) takes places 'as is', without consideration of the computational status of the argument, but only if the term under consideration is a function application itself. The second is similar, but the main distinction is that it forces the evaluation of the argument to a value before the main redex gets contracted. CBN-reduction is normalising, so CBN-reduction will lead to a CBN-normal form, if it exists. The main advantage of CBV is that, since an argument is evaluated before being substituted, unnecessary duplication of reduction is avoided, but at the price that this strategy is not normalising, so CBV-reduction does not necessarily lead to a normal form.

These two reduction strategies differ significantly in semantics and for example do not share all normal forms, and it is an interesting question if a calculus can be found in which both can be modelled coherently, preserving typeability, assignable types, denotational semantics, etc. In [11] Levy proposed the calculus Call by Push Value (CBPV) as answer to this question, as a subsuming paradigm for the CBN and CBV paradigms of the λ-calculus.

CBPV is presented with a deterministic reduction relation, that can be thought of as encoded in the term itself, as reduction is blocked and activated through additional syntax. As is usual, since their reduction relation is different, the CBN and CBV reduction strategies of the λ-calculus are interpreted differently into CBPV; these interpretations preserve the operational and denotational semantics, making CBPV a good candidate for the study of both.

However, CBPV is not the only calculus that achieves the embodiment of this duality. Herbelin and Curien [7, 3] defined the calculus $\overline{\lambda}\mu\tilde{\mu}$ that represents proofs and cut-elimination of a variant Gentzen's Sequent Calculus for Classical Logic [6] with focus, which can be seen as a generalisation and extension of Parigot's $\lambda\mu$-calculus [13], and has been shown to represent an abstract machine which models both these paradigms successfully. In particular, also the

interpretations of the CBN and CBV λ-calculus into $\overline{\lambda}\mu\tilde{\mu}$ preserve the reductions. Moreover, [11] defines an abstract machine, CK, to give an operational semantics for CBPV, which very closely corresponds to the workings of these interpretations into $\overline{\lambda}\mu\tilde{\mu}$.

This observation lead us to investigate the link between these two calculi - $\overline{\lambda}\mu\tilde{\mu}$ and CBPV - and here we will give an interpretation of CBPV into $\overline{\lambda}\mu\tilde{\mu}$, inspired by the behaviour of CBPV in the abstract machine CK [12]. Using the motivation for $\overline{\lambda}\mu\tilde{\mu}$ as an abstract machine to run the λ-calculus in, we attempt to use a similar technique for translating CBPV. However, the artificial halting and continuing of computations - 'thunking' and 'forcing' - cannot be modelled by the traditional $\overline{\lambda}\mu\tilde{\mu}$ syntax. A computation M that is 'thunked' and then immediately 'forced' continues running as M, removing the surrounding 'thunk' and 'force' syntax. In the abstract machine CK, this computational step doesn't interact with the stack at all, but still reduces. So it cannot be represented as a function or a variable, hence cannot be translated into $\overline{\lambda}\mu\tilde{\mu}$ syntax. To overcome this, we basically remove the 'thunk' - 'force' reduction step, for which we had to change the notion of term substitution.

Outline of this paper

We start in Section 1 by giving a quick overview of the λ-calculus, and its relevant reduction strategies; this is followed in Section 2 where we revisit the functional core of Levy's CBPV-calculus and the abstract machine CK that is used to give an operational semantics for CBPV. In Section 3 we will revisit some results on the interpretation of CBN-λ and CBV-λ into CBPV, as well as those presented in λ-*val* [5]; we will see that not all results shown are as powerful as first suggested. For example, the interpretations do not respect reduction, only equality.

To address the shortcomings we observe in Section 3, in Section 4 we will present *essential* CBPV, a variant of CBPV where only variables can be forced, and only 'real' computations can be thunked (so not forced variables). This is paired with a different notion of term substitution, that carefully avoids to create a term like $\{M\}!$ since these are no longer accepted as terms. We will define interpretations of CBN-λ and CBV-λ into eCBPV and show that now reduction is respected. But we achieve more: because of this change, we can also establish a strong relation with $\overline{\lambda}\mu\tilde{\mu}$ and show the interpretation results in Section 6.

In Section 5 we will give a short overview of Herbelin's calculus $\overline{\lambda}\mu\tilde{\mu}$ with a more detailed definition of its CBN and CBV-reduction *strategies*, so more detailed than done in previous papers. We define a single interpretation of λ-terms into $\overline{\lambda}\mu\tilde{\mu}$ (so do *not* define a CBN or CBN-interpretation), and show through detailed proofs that CBN-λ is respected under that interpretation by CBN-$\overline{\lambda}\mu\tilde{\mu}$, and that CBV-$\lambda$ is respected under that interpretation by CBV-$\overline{\lambda}\mu\tilde{\mu}$. This is followed in Section 6 by the definition of an interpretation of eCBPV-terms into $\overline{\lambda}\mu\tilde{\mu}$, and one for the abstract machine CK into $\overline{\lambda}\mu\tilde{\mu}$, and show that both fully preserve reduction, and that the first also preserves typeability. In fact, reduction in eCBPV is fully modelled in CBN-$\overline{\lambda}\mu\tilde{\mu}$, emphasising again that CBPV is, in fact, a call-by-name calculus.

1 The λ-calculus

We assume the reader to be familiar with the λ-calculus; we just recall the definition of λ-terms and β-reduction.

Definition 1.1 (LAMBDA TERMS, CBN AND CBV REDUCTION [2]) *i*) *λ-terms* are defined by the grammar:

$$\begin{array}{rcl} M,N & ::= & V \mid MN \\ V & ::= & x \mid \lambda x.M \quad (\textit{values}) \end{array}$$

ii) (One-step) β-reduction is defined using the β-rule

$$(\beta):\ (\lambda x.M)\,N \to M[N/x]$$

where $[N/x]$ stands for the implicit substitution[1] that is to take place immediately and silently, and evaluation contexts that are defined as terms with a single hole by:

$$\mathsf{C} \ ::= \ \lceil\ \rfloor^{2} \mid \mathsf{C}M \mid M\mathsf{C} \mid \lambda x.\mathsf{C}$$

We write $\mathsf{C}[M]$ for the term obtained from the context C by replacing its hole $\lceil\ \rfloor$ with M, allowing variables to be captured. One-step β reduction is defined as the compatible closure of the β-rule through:

$$(\beta):\ \mathsf{C}[(\lambda x.M)\,N] \ \to \ \mathsf{C}[M[N/x]]$$

for any evaluation context. We write $\to_\beta^*$ for the transitive closure of $\to_\beta$, and use that notation for all the notions of reduction we consider in this paper.

iii) *Call-by-name evaluation contexts* are defined through:

$$\mathsf{C}_{\mathbf{N}} \ ::= \ \lceil\ \rfloor \mid \mathsf{C}_{\mathbf{N}}M$$

*Call-by-name (*CBN*) reduction*, $\to_{\mathbf{N}}$ is defined through:

$$\mathsf{C}_{\mathbf{N}}[(\lambda x.M)\,N] \ \to \ \mathsf{C}_{\mathbf{N}}[M[N/x]]$$

(also known as *lazy* reduction)

iv) *Call-by-value evaluation contexts* are defined through:

$$\mathsf{C}_{\mathbf{V}} \ ::= \ \lceil\ \rfloor \mid \mathsf{C}_{\mathbf{V}}M \mid V\,\mathsf{C}_{\mathbf{V}}$$

*Call-by-value (*CBV*) reduction*, $\to_{\mathbf{V}}$, is defined through: $\mathsf{C}_{\mathbf{V}}[(\lambda x.M)\,V] \ \to \ \mathsf{C}_{\mathbf{V}}[M[V/x]]$

For the discourse of this paper, it is important to highlight the difference between CBN and CBV-reduction.

Remark 1.2 Let $M \equiv M_1\,M_2\,M_3\cdots M_n$ be a λ-term such that M_1 is not an application, then either $M_1 \equiv \lambda x.M_1'$, or $M_1 \equiv y$. If $M_1 \equiv \lambda x.N$, then CBN-reduction on M will contract the redex $(\lambda x.M_1')\,M_2$ to obtain the term $M_1'[M_2/x]\,M_3\cdots M_n$, so without touching M_2. On the other hand, CBV-reduction will first run M_2 using the CBV-reduction strategy until it reaches an abstraction or variable V_2; only then will the first redex become contractable, and will M reduce to $M_1'[V_2/x]\,M_3\cdots M_n$.

Curry (or simple) type assignment for the λ-calculus is defined as follows:

Definition 1.3 (CURRY TYPE ASSIGNMENT FOR THE λ-CALCULUS) *i*) Let φ range over a countable (infinite) set of type-variables. The set of *Curry types* is defined by the grammar:

$$A, B \ ::= \ \varphi \mid A \to B$$

ii) A *context of variables* Γ is a partial mapping from term variables to types, denoted as a finite set of *statements* $x{:}A$, such that the *subjects* of the statements (x) are distinct.

iii) *Curry type assignment* is defined by the following inference system:

$$(Ax):\ \frac{}{\Gamma, x{:}A \vdash x : A} \qquad (\to I):\ \frac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}\ (x \notin \Gamma) \qquad (\to E):\ \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

Type assignment in this system is decidable, and it enjoys a Curry-Howard correspondence

[1] The notation $\{N/x\}$ is traditionally used, but since the curly brackets are used for term construction in CBPV, we decided to use a different notation here.

[2] The notation for the 'hole' in contexts differs in the literature; we go for the non-standard $\lceil\ \rfloor$, since the alternatives, like $[\,]$, or $\{\,\}$, or $(\,)$, are used for different things in this paper.

with implicative intuitionistic logic.

2 Call By Push Value

Call by Push Value, presented by [11], is a calculus designed to make the execution order of a λ-based term explicit, by presenting a deterministic reduction system, but one that is capable of expressing both CBN-λ and CBV-λ.

CBPV, as does CBV-λ, splits its terms into two distinct categories: *computations* and *values*, but does that in a different way, in that abstraction no longer yields a value, only the construct of 'thunking' (blocking reductions) a term does. The slogan from [11] is

"a value is, a computation does"

and this is expressed through the reduction relation, where only computations reduce to each other, and values are substituted or parameters. A computation can be used as a value if it is 'thunked', and a value can be used as a computation if it is 'forced', unblocking the reductions. This arbitrarily halting and continuing of computation can easily be brought back to just the essential steps, as we will see in this paper. Computations include forced values, λ-abstractions, applications, sequencing, and returning, and values are variables and thunked computations. Sequencing and returning play a major role in the encoding of CBV-λ, to influence the order of evaluation in an application. Under CBV, the subterm N in $(\lambda x.\, M)\, N$ gets evaluated first, but in CBPV an application is always of the shape PV, which does not allow for reduction of the argument; the encoding fixes this, mainly through sequencing. In fact, its name notwithstanding, reduction in CBPV is mainly 'by name' (or lazy) in nature; this is reflected in the ease with which CBN-λ can be encoded. Its type system has a Curry-Howard-style correspondence with intuitionistic linear logic [14], hence only intuitionistic calculi can be translated into CBPV.

In this paper we use the syntax from [5] but for reasons of simplicity consider pure, functional CBPV without effects - in particular, we do not the consider sum and product types[3] that were included in the original definition of CBPV. Also, Figure 1 of [11] presents CBPV as a language of typeable terms, with an operational semantics using the abstract machine CK (see Definition 2.7), and equality only defined for typeable terms in Figure 2; since normally typeability implies termination, these results are rather limited. Here we will deviate from that approach, and treat CBPV as a pure λ-calculus: we will define syntax and type assignment separately, as well as a one-step reduction relation on terms. This will enable us to interpret the untyped CBN/CBV-λ-calculus, but more importantly we can deal with non-terminating reductions as well.

Definition 2.1 (CBPV TERMS) There are two categories of CBPV-terms: values (ranged over by V, W) and computations (ranged over by M, N). They are defined through the grammar:

$$\begin{array}{lll} V, W & ::= & x \mid \{M\} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{(values)} \\ M, N & ::= & V! \mid \lambda x.\, M \mid MV \mid \mathit{ret}\, V \mid x := M; N \qquad\qquad \textit{(computations)} \end{array}$$

The notion of free and bound variables is defined as usual, taking x to be bound in the terms $\lambda x.\, M$ and $x := M; N$; in the latter, the occurrences of x in N are bound, and by Barendregt's convention x does not occur in M.

[3] This paper ignores the values $()$, (V_1, V_2) and $\mathtt{inj}_i V$, and the computations $\mathtt{split}(V, x_1.x_2.M)$, case constructs $\mathtt{case}_0(V)$, $\langle\rangle$, $\mathtt{case}(V, x_1.M_1, x_2.M_2)$, pairs $\langle M_1, M_2\rangle$ and projections $\mathtt{prj}_i M$. These are all valuable features for programming languages, but in the context of CBPV do not themselves cause any non-standard treatment, and greatly complicate the presentation of the calculus. Moreover, we would have to add those features to $\overline{\lambda}\mu\tilde{\mu}$ as well.

We read $\{\cdot\}$ as 'thunk', which represents the blocking of a computation by making it a value, and $\cdot!$ as 'force', which pushes a value into becoming a computation. Notice that all variables are values, and hence they can only be replaced by computations if said computation is 'thunked' (*i.e.* of the form $\{M\}$).

Many different notations have been used for CBPV in the past. For example, application MV is written 'operand first' as $V`M$ in [10] and past papers used *let* $x = M$ *in* N or even *let* $x \leftarrow M$ *in* N for our $x := M; N$. We prefer to use the latter notation since it immediately suggest the correct reduction order: to effectuate an assignment $x := M$, we first evaluate M to completion (so until it becomes of the shape $\mathit{ret}\, V$) and then assign its result to x; then, when evaluating N, whenever we encounter x, the result of M is used.[4]

Reduction is not defined as a one-step reduction relation in [10], but only through a big-step semantics in Figure 4; [11] defines a typeable equational theory, so equality is only defined on typeable terms. We have extracted one-step reduction from that.

Definition 2.2 (CBPV REDUCTION RULES) *i*) The three basic reduction rules are defined by:

$$\begin{array}{rrcll} (C)^5: & (\lambda x.M)\,V & \to & M[V/x] & (\mathit{contract}) \\ (U): & \{M\}! & \to & M & (\mathit{unblock}) \\ (F): & x := \mathit{ret}\, V; M & \to & M[V/x] & (\mathit{force}) \end{array}$$

ii) The CBPV-evaluation contexts are defined through:

$$\mathsf{C} \;::=\; \ulcorner\, \lrcorner \mid \mathsf{C}V \mid x := \mathsf{C}; M$$

iii) CBPV-reduction is defined through:

$$M \to N \;\Rightarrow\; \mathsf{C}[M] \to \mathsf{C}[N]$$

whenever $M \to N$ through either rule (C), (U), or (F). We will use $\to_{\mathrm{P}}$ for this notion of reduction.

Notice that reduction is not allowed to take place under an abstraction, irrespective of calling an abstraction a computation, not a value: the terminology is confusing.

It might be worthwhile to observe that the notion of 'value' in CBPV is not the standard one. Normally, a value is either a variable or an abstraction, and CBV-reduction stands for the strategy that arguments N in an application MN are only ever reduced when the running of M produces a value; then N will be reduced until it itself becomes a value, and the redex $(\lambda x.M)\,N$ will only ever be contracted if N is a value - the argument is called 'by value'. This is not the case for CBPV. Here values are variables or blocked computations of the shape $\{M\}$, where M can be any computation. Arguments are never reduced, and a redex $(\lambda x.M)\,\{N\}$ can be contracted even when N is an application.

Example 2.3 We have

$$(\lambda x.x!)\,\{y!y\} \;\to_{\mathrm{P}}\; \{y!y\}! \;\to_{\mathrm{P}}\; y!y$$

whereas the corresponding λ-term $(\lambda x.x)\,(yy)$ is in CBV-normal form.

As unforced variables can only occur on the right-hand side of an application, they can never be the result of a computation, as is evidenced by $(\lambda x.x!)\,y \to y!$.

There are also infinite reductions:

[4] Traditionally the *let*-construct *let* $x = M$ *in* N represents *sequentiality* since in case x does not occur in N, the *let*-expression is equivalent to $M; N$. However, we should perhaps point out that in a language without side-effects, as we are considering here, the term $M;\ N$ stands for just N.

[5] This rule is called $(\to)$ in [10].

$$
\begin{array}{ll}
(\lambda x.(x!)x)\,\{\lambda x.(x!)x\} & \to_{\mathrm{P}}\ (C)\\
(\{\lambda x.(x!)x\}!)\,\{\lambda x.(x!)x\} & \to_{\mathrm{P}}\ (U)\\
(\lambda x.(x!)x)\,\{\lambda x.(x!)x\} &
\end{array}
$$

The notion of type assignment for CBPV is atypical in that values are typed with value types, and computations with computation types.

Definition 2.4 (CBPV TYPE ASSIGNMENT) *i*) There are two sorts of types for CBPV terms:

$$
\begin{array}{lll}
A, B & ::= & \varphi \mid \mathsf{U}\mathsf{A} \qquad (\textit{value types})\\
\mathsf{A}, \mathsf{B} & ::= & A \to \mathsf{B} \mid \mathsf{F}A \qquad (\textit{computation types})
\end{array}
$$

ii) Contexts are defined as in Definition 1.2.

iii) Type assignment for CBPV is defined through the following inference system; it introduces two notions, $\vdash_{\mathrm{V}}$ and $\vdash_{\mathrm{C}}$, that express a judgement for a value or computation, respectively.

$$
(\textit{axiom}):\ \frac{}{\Gamma, x{:}A \vdash_{\mathrm{V}} x : A} \qquad (\textit{thunk}):\ \frac{\Gamma \vdash_{\mathrm{C}} M : \mathsf{A}}{\Gamma \vdash_{\mathrm{V}} \{M\} : \mathsf{U}\,\mathsf{A}}
$$

$$
(\textit{abstr}):\ \frac{\Gamma, x{:}A \vdash_{\mathrm{C}} M : \mathsf{B}}{\Gamma \vdash_{\mathrm{C}} \lambda x.M : A \to \mathsf{B}} \qquad (\textit{appl}):\ \frac{\Gamma \vdash_{\mathrm{C}} M : A \to \mathsf{B} \quad \Gamma \vdash_{\mathrm{V}} V : A}{\Gamma \vdash_{\mathrm{C}} MV : \mathsf{B}}
$$

$$
(\textit{force}):\ \frac{\Gamma \vdash_{\mathrm{V}} V : \mathsf{U}\,\mathsf{A}}{\Gamma \vdash_{\mathrm{C}} V! : \mathsf{A}} \qquad (\textit{ret}):\ \frac{\Gamma \vdash_{\mathrm{V}} V : A}{\Gamma \vdash_{\mathrm{C}} \mathit{ret}\,V : \mathsf{F}\,A} \qquad (\textit{assign}):\ \frac{\Gamma \vdash_{\mathrm{C}} M : \mathsf{F}A \quad \Gamma, x{:}A \vdash_{\mathrm{C}} N : \mathsf{B}}{\Gamma \vdash_{\mathrm{C}} x := M; N : \mathsf{B}}
$$

We can show that this notion of type assignment is sound, *i.e.* satisfies subject-reduction.

Lemma 2.5 *i) If* $\Gamma, x{:}A \vdash_{\mathrm{V}} V : B$, *and* $\Gamma \vdash_{\mathrm{V}} V' : A$, *then* $\Gamma \vdash_{\mathrm{V}} V[V'/x] : B$.
ii) If $\Gamma, x{:}A \vdash_{\mathrm{C}} M : \mathsf{B}$, *and* $\Gamma \vdash_{\mathrm{V}} V' : A$, *then* $\Gamma \vdash_{\mathrm{V}} M[V'/x] : \mathsf{B}$.

Proof: By simultaneous induction on the definition of '$\cdot\,[V/x]$'. □

Theorem 2.6 (SUBJECT REDUCTION) *If* $\Gamma \vdash_{\mathrm{C}} M : \mathsf{B}$, *and* $M \to_{\mathrm{P}} N$, *then* $\Gamma \vdash_{\mathrm{C}} N : \mathsf{B}$.

Proof: By induction on the definition of reduction; we will only consider the base cases.

(C): Then $M = (\lambda x.M')\,V$, and there exists A such that $\Gamma, x{:}A \vdash_{\mathrm{C}} M' : \mathsf{B}$ and $\Gamma \vdash_{\mathrm{V}} V : A$; the result follows from Lemma 2.5.

(U): Then $M = \{M'\}!$, and we have $\Gamma \vdash_{\mathrm{C}} M' : \mathsf{B}$ by rules (*thunk*) and (*force*).

(F): Then $M = x := \mathit{ret}\,V; M'$ and there exists A such that $\Gamma, x{:}A \vdash_{\mathrm{C}} M' : \mathsf{B}$ and $\Gamma \vdash_{\mathrm{V}} V : A$; the result follows from Lemma 2.5. □

In the same vein as Krivine's machine for the λ-calculus [9], in [10] Levy defines an operational semantics of CBPV in CK, a stack machine. The machine uses configurations $\langle M \mid S \rangle$ where M is the computation being evaluated and S is the environment (a stack of values, in combination with contexts that are awaiting the completion of the evaluation of a term to be inserted in the context) in which the evaluation of M takes place. We will present a variant of that machine here.

Definition 2.7 (CK-MACHINE [10]) *i*) *Evaluation stacks* are defined through:

$$
S \ ::= \ \epsilon \mid V : S \mid x := [\,]; M : S
$$

ii) The evaluation of configurations $\langle M \mid S \rangle$ of M in the evaluation stack S is defined by:

$$\begin{array}{rcl}
\langle \lambda x.M \,|\, V : S\rangle & \to_{\text{CK}} & \langle M[V/x] \,|\, S\rangle \\
\langle MV \,|\, S\rangle & \to_{\text{CK}} & \langle M \,|\, V : S\rangle \\
\langle x := M; N \,|\, S\rangle & \to_{\text{CK}} & \langle M \,|\, x := [\,]; N : S\rangle \\
\langle \mathit{ret}\, V \,|\, x := [\,]; M : S\rangle & \to_{\text{CK}} & \langle M[V/x] \,|\, S\rangle \\
\langle \{M\}! \,|\, S\rangle & \to_{\text{CK}} & \langle M \,|\, S\rangle
\end{array}$$

We define $=_{\text{CK}}$ as the equivalence relation generated by $\to_{\text{CK}}$.

iii) We define an interpretation for CBPV to CK by: $[\![M]\!]^{\text{CK}} = \langle M \,|\, \epsilon\rangle$.

Notice that in the last step, there is no interaction with the stack.

We can now show that CBPV-reduction is preserved under $=_{\text{CK}}$.

Theorem 2.8 *If $M \to_{\text{P}} N$, then $\langle M \,|\, S\rangle =_{\text{CK}} \langle N \,|\, S\rangle$.*

Proof: By induction on the definition of $\to_{\text{P}}$.

$((\lambda x.M)\, V \to_{\text{P}} M[V/x])$: $\langle (\lambda x.M)\, V \,|\, S\rangle \to_{\text{CK}} \langle \lambda x.M \,|\, V : S\rangle \to_{\text{CK}} \langle M[V/x] \,|\, S\rangle$

$(\{M\}! \to_{\text{P}} M)$: $\langle \{M\}! \,|\, S\rangle \to_{\text{CK}} \langle M \,|\, S\rangle$

$(x := \mathit{ret}\, V; M \to_{\text{P}} M[V/x])$: $\langle x := \mathit{ret}\, V; M \,|\, S\rangle \to_{\text{CK}} \langle \mathit{ret}\, V \,|\, x := [\,]; M : S\rangle \to_{\text{CK}}$
$\langle M[V/x] \,|\, S\rangle$

$(M \to_{\text{P}} N \Rightarrow MV \to_{\text{P}} NV)$: $\langle MV \,|\, S\rangle \to_{\text{CK}} \langle M \,|\, V : S\rangle =_{\text{CK}} (IH)\ \langle N \,|\, V : S\rangle \leftarrow_{\text{CK}} \langle NV \,|\, S\rangle$

$(M \to_{\text{P}} N \Rightarrow x := M; P \to_{\text{P}} x := N; P)$: $\langle x := M; P \,|\, S\rangle \to_{\text{CK}} \langle M \,|\, x := [\,]; P : S\rangle =_{\text{CK}} (IH)$
$\langle N \,|\, x := [\,]; P : S\rangle \leftarrow_{\text{CK}} \langle x := N; P \,|\, S\rangle$ □

With this result, we immediately have that CBPV-reduction is preserved by the interpretation $[\![\cdot]\!]^{\text{CK}}$.

Corollary 2.9 *If $M \to_{\text{P}}^* N$, then $[\![M]\!]^{\text{CK}} =_{\text{CK}} [\![N]\!]^{\text{CK}}$.*

The last property is claimed in [10] (adapted here to our notation) as 'For any closed computation M, we have $M \Downarrow T$ iff $\langle M \,|\, \epsilon\rangle \to_{\text{CK}}^* \langle T \,|\, \epsilon\rangle$'. However, this seems unlikely, since we have:

$(M \to_{\text{P}} N \Rightarrow MV \to_{\text{P}} NV)$: $\langle MV \,|\, S\rangle \to_{\text{CK}} \langle M \,|\, V : S\rangle \to_{\text{CK}} (IH)\ \langle N \,|\, V : S\rangle$

Now $\langle N \,|\, V : S\rangle$ does not reduce under $\to_{\text{CK}}$ to $\langle NV \,|\, S\rangle$, rather we only have the reverse: $\langle NV \,|\, S\rangle \to_{\text{CK}} \langle N \,|\, V : S\rangle$; hence our use of $=_{\text{CK}}$.

We will see this problem come back in Theorem 3.3.

3 Interpreting the λ-calculus in CBPV

There are two different ways to interpret the λ-calculus - by modelling CBN reduction or modelling CBV reduction. It is claimed that the syntax of CBPV allows us to explicitly encode the execution order, so we will encode our chosen reduction strategy.

Levy [10] presents what he calls Fine-Grain CBV, a typed λ-calculus extended with two `let` constructs and a conditional construct, presented as proofs; a variant of that calculus is considered in [5], but there with product and sum. Since in this paper we are mainly interested in the relation between the pure λ-calculus, CBPV, and $\overline{\lambda}\mu\tilde{\mu}$, all untyped, we will concentrate on the pure functional component of those calculi.

3.1 A CBV-interpretation of Λ in CBPV

Rather than following the path of [10, 5] where extended calculi are studied, we will here focus on the interpretation of the pure λ-calculus with CBV-reduction, as defined in Definition 1.1, in CBPV, so start from the thought that the CBV is a *reduction strategy* for the λ-calculus that does not require changing the syntax. We will revisit the results of [10, 5].

Definition 3.1 ([10]) Levy's CBV-interpretation $\llbracket \cdot \rrbracket^{\lambda}_{\mathrm{v}}$ of λ-terms into CBPV is defined through:

$$\begin{array}{rcl} \llbracket x \rrbracket^{\lambda}_{\mathrm{v}} & = & ret\, x \\ \llbracket \lambda x.\, M \rrbracket^{\lambda}_{\mathrm{v}} & = & ret\, \{\lambda x.\, \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}\} \\ \llbracket MN \rrbracket^{\lambda}_{\mathrm{v}} & = & x := \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; x!y \end{array}$$

Notice the use of sequencing. We will show that this interpretation preserves reduction and assignable types; we first show that it respects term-substitution.

Lemma 3.2 (SUBSTITUTION LEMMA FOR $\llbracket \cdot \rrbracket^{\lambda}_{\mathrm{v}}$) *i)* $\llbracket M \rrbracket^{\lambda}_{\mathrm{v}} [w/z] = \llbracket M [w/z] \rrbracket^{\lambda}_{\mathrm{v}}$.
ii) $\llbracket M \rrbracket^{\lambda}_{\mathrm{v}} [\{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z] = \llbracket M [\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}$.

Proof: i) By straightforward induction on the definition of substitution.
ii) $(M \equiv z)$: $\llbracket z \rrbracket^{\lambda}_{\mathrm{v}} [\{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z] = (ret\, z) [\{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z] = ret\, \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\} = \llbracket \lambda w.R \rrbracket^{\lambda}_{\mathrm{v}} = \llbracket z [\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}$

$(M \equiv u \text{ and } u \neq z)$: $\llbracket u \rrbracket^{\lambda}_{\mathrm{v}} [\{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z] = (ret\, u) [\{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z] = ret\, u = \llbracket u \rrbracket^{\lambda}_{\mathrm{v}} = \llbracket u [\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}$

The other cases follow by induction. □

With this result we can show that the interpretation $\llbracket \cdot \rrbracket^{\lambda}_{\mathrm{v}}$ respects reduction up to equality.

Theorem 3.3 *If* $M \to_{\mathrm{v}} N$, *then* $\llbracket M \rrbracket^{\lambda}_{\mathrm{v}} =_{\mathrm{p}} \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}$.

Proof: $((\lambda z.\, M)\, w \to_{\mathrm{v}} M[w/z])$: $\llbracket (\lambda z.\, M)\, w \rrbracket^{\lambda}_{\mathrm{v}} \triangleq x := \llbracket \lambda z.\, M \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket w \rrbracket^{\lambda}_{\mathrm{v}}; x!y \triangleq x := ret\, \{\lambda z.\, \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}\}; y := ret\, w; x!y \to_{\mathrm{p}} y := ret\, w; \{\lambda z.\, \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}\}!y \to_{\mathrm{p}} \{\lambda z.\, \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}\}!w \to_{\mathrm{p}} (\lambda z.\, \llbracket M \rrbracket^{\lambda}_{\mathrm{v}})\, w \to_{\mathrm{p}} \llbracket M \rrbracket^{\lambda}_{\mathrm{v}} [w/z] = (3.2)\ \llbracket M [w/z] \rrbracket^{\lambda}_{\mathrm{v}}$

$((\lambda z.\, M)\, (\lambda w.R) \to_{\mathrm{v}} M[\lambda w.R/z])$: $\llbracket (\lambda z.\, M)\, (\lambda w.R) \rrbracket^{\lambda}_{\mathrm{v}} \triangleq x := \llbracket \lambda z.\, M \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket \lambda w.R \rrbracket^{\lambda}_{\mathrm{v}}; x!y \triangleq x := ret\, \{\lambda z.\, \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}\}; y := ret\, \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}; x!y \to_{\mathrm{p}} y := ret\, \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}; \{\lambda z.\, \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}\}!y \to_{\mathrm{p}} \{\lambda z.\, \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}\}!\{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\} \to_{\mathrm{p}} (\lambda z.\, \llbracket M \rrbracket^{\lambda}_{\mathrm{v}})\, \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\} \to_{\mathrm{p}} \llbracket M \rrbracket^{\lambda}_{\mathrm{v}} [\{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z] = (3.2)\ \llbracket M [\{\lambda w.R\}/z] \rrbracket^{\lambda}_{\mathrm{v}}$

$(M \to_{\mathrm{v}} N \Rightarrow MP \to_{\mathrm{v}} NP)$: $\llbracket MP \rrbracket^{\lambda}_{\mathrm{v}} \triangleq x := \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; x!y =_{\mathrm{p}} (IH)$
$x := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; x!y \triangleq \llbracket NP \rrbracket^{\lambda}_{\mathrm{v}}$

$(M \to_{\mathrm{v}} N \Rightarrow zM \to_{\mathrm{v}} zN)$: $\llbracket zM \rrbracket^{\lambda}_{\mathrm{v}} \triangleq x := ret\, z; y := \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}; x!y \to_{\mathrm{p}} y := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; z!y$
and $\llbracket zN \rrbracket^{\lambda}_{\mathrm{v}} \triangleq x := z; y := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; x!y \to_{\mathrm{p}} y := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; z!y$

$(M \to_{\mathrm{v}} N \Rightarrow (\lambda w.R)\, M \to_{\mathrm{v}} (\lambda w.R)\, N)$: $\llbracket (\lambda w.R)\, M \rrbracket^{\lambda}_{\mathrm{v}} \triangleq x := ret\, \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}; y := \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}; x!y \to_{\mathrm{p}} y := \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}; \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}!y =_{\mathrm{p}} (IH)\ y := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}!y$
and $\llbracket (\lambda w.R)\, N \rrbracket^{\lambda}_{\mathrm{v}} \triangleq x := ret\, \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}; y := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; x!y \to_{\mathrm{p}} y := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}!y$ □

Notice that, in the last two cases, as was also the case or $\to_{\mathrm{CK}}$, the two interpretations of the terms involved in the $\to_{\mathrm{v}}$-reduction step are not related through reduction, but through equality.

We can show that the systems are related also on the level of types. We first define an interpretation of types.

Definition 3.4 ([10]) The CBV-interpretation of Curry types into CBPV-types is defined through:

$$\begin{array}{rcl} \overline{\varphi} & \triangleq & \varphi \\ \overline{A \to B} & \triangleq & \mathsf{U}(\overline{A} \to \mathsf{F}\overline{B}) \end{array}$$

This interpretation is straightforwardly extended to contexts: $\overline{\Gamma} \triangleq \{ x{:}\overline{A} \mid x{:}A \in \Gamma \}$.

We can now show:

Lemma 3.5 *If* $\Gamma \vdash M : A$, *then* $\overline{\Gamma} \vdash_{\mathrm{C}} \llbracket M \rrbracket^{\lambda}_{\mathrm{V}} : \mathsf{F}\,\overline{A}$.

Proof: Simultaneously by induction on definition of type assignment. We only show the base case, the other follow by induction.

$((Ax))$: Then $v \equiv x$, and $\Gamma = \Gamma', x{:}A$; since $\llbracket v \rrbracket^{\lambda}_{\mathrm{V}} = x$, and $\overline{\Gamma} = \overline{\Gamma'}, x{:}\overline{A}$. We can construct:

$$\dfrac{\dfrac{}{\overline{\Gamma} \vdash_{\mathrm{V}} x : \overline{A}}\ (axiom)}{\overline{\Gamma} \vdash_{\mathrm{V}} ret\, x : \mathsf{F}\,\overline{A}}\ (ret)$$

Notice that $\llbracket x \rrbracket^{\lambda}_{\mathrm{V}} \triangleq ret\, x$. □

3.2 A CBN-handling of Λ in CBPV

In [10], Levy also deals with mapping CBN-reduction for the λ-calculus into CBPV, but this attempt is less successful than the one dealing with CBV; [5] basically repeats Levy's work and result.

Under CBN-reduction in the λ-calculus, the right-hand-side of an application is never evaluated; if we think of CBN reductions as CBPV computations, those parts of terms need to be 'thunked' (or halted), and only resume after being substituted for a variable, so a variable should be 'forced'. Using this intuition, we can revisit Levy's CBN-treatment of λ-terms into CBPV.

Definition 3.6 (Simulation of CBN [10, 5]) The relation $\mapsto_{\mathrm{N}}$ between λ-terms and CBPV-terms is defined as:

$$\dfrac{}{x \mapsto_{\mathrm{N}} x!} \qquad \dfrac{M \mapsto_{\mathrm{N}} M'}{\lambda x.M \mapsto_{\mathrm{N}} \lambda x.M'} \qquad \dfrac{M \mapsto_{\mathrm{N}} M' \quad N \mapsto_{\mathrm{N}} N'}{M'\{N'\}} \qquad \dfrac{M \mapsto_{\mathrm{N}} M'}{M \mapsto_{\mathrm{N}} \{M'\}!}\ ^{6}$$

Notice that '$x := N; M'$' and '$ret\, V$' are not used here.

A problem with this definition is that it does not define an interpretation; the fourth rule allows to place an arbitrary amount of 'force'-'thunk' pairs around interpreted λ-terms, so for every λ-term M there are infinitely many CBPV-terms M' such that $M \mapsto_{\mathrm{N}} M'$. Thereby this relation does not give anything close to a semantics. [5] states that $\mapsto_{\mathrm{N}}$ is injective, but that seems to be an unnecessary claim for a relation that follows the application-abstraction structure of terms precisely, and is not even a function.

Notice that the fourth rule effectively models the reduction rule (U) in reverse, arbitrarily many times; its use is even allowed under abstraction, thereby stepping outside the CBPV reduction strategy.

We cannot show that reduction is preserved under this relation.

Example 3.7 Remark that $(\lambda x.xx)\,(\lambda x.xx) \to_{\mathrm{N}} (\lambda x.xx)\,(\lambda x.xx)$. We have:

[6] The last rule is missing from Figure 3 in the Appendix of [5], but we assume this is in error.

$$\dfrac{\dfrac{\dfrac{\overline{x \mapsto_{\mathrm{N}} x!} \quad \overline{x \mapsto_{\mathrm{N}} x!}}{xx \mapsto_{\mathrm{N}} x!\{x!\}}}{\lambda x.xx \mapsto_{\mathrm{N}} \lambda x.x!\{x!\}} \quad \dfrac{\dfrac{\overline{x \mapsto_{\mathrm{N}} x!} \quad \overline{x \mapsto_{\mathrm{N}} x!}}{xx \mapsto_{\mathrm{N}} x!\{x!\}}}{\lambda x.xx \mapsto_{\mathrm{N}} \lambda x.x!\{x!\}}}{(\lambda x.xx)\,(\lambda x.xx) \mapsto_{\mathrm{N}} (\lambda x.x!\{x!\})\{\lambda x.x!\{x!\}\}}$$

and

$$\begin{array}{rll} (\lambda x.x!\{x!\})\{\lambda x.x!\{x!\}\} & \to_{\mathrm{P}}(C) & x!\{x!\}\,[\{\lambda x.x!\{x!\}\}/x] \\ & = & \{\lambda x.x!\{x!\}\}!\{\{\lambda x.x!\{x!\}\}!\} \\ & \to_{\mathrm{P}}(U) & (\lambda x.x!\{x!\})\{\{\lambda x.x!\{x!\}\}!\} \end{array}$$

It will be clear that $(\lambda x.x!\{x!\})\{\lambda x.x!\{x!\}\}$ does not run to itself. This is partially solved by the fourth rule, since we can also show:

$$\dfrac{\dfrac{\dfrac{\overline{x \mapsto_{\mathrm{N}} x!} \quad \overline{x \mapsto_{\mathrm{N}} x!}}{xx \mapsto_{\mathrm{N}} x!\{x!\}}}{\lambda x.xx \mapsto_{\mathrm{N}} \lambda x.x!\{x!\}} \quad \dfrac{\dfrac{\overline{x \mapsto_{\mathrm{N}} x!} \quad \overline{x \mapsto_{\mathrm{N}} x!}}{xx \mapsto_{\mathrm{N}} x!\{x!\}}}{\lambda x.xx \mapsto_{\mathrm{N}} \{\lambda x.x!\{x!\}\}!}}{(\lambda x.xx)\,(\lambda x.xx) \mapsto_{\mathrm{N}} (\lambda x.x!\{x!\})\{\{\lambda x.x!\{x!\}\}!\}}$$

One could argue that it could have been better to reverse the above idea and add the rule (U) without limitation to solve this quandary; notice that then we would have:

$$(\lambda x.x!\{x!\})\{\{\lambda x.x!\{x!\}\}!\} \;\to_{\mathrm{P}}\; (\lambda x.x!\{x!\})\{\lambda x.x!\{x!\}\}$$

as desired, and would have obtained a CBN-interpretation.

It is possible to show that $\mapsto_{\mathrm{N}}$ respects type assignment; this result is very much like that of Theorem 4.12, and we therefore will not present it here.

Using this notion, [10, 5] show:

Lemma 3.8 ([10, 5]) *i) (Forwards simulation) Let M, N be λ-terms. If $M \mapsto_{\mathrm{N}} Q$, and $M \to_{\mathrm{N}}^{*} N$, then there exists a* CBPV *term R such that $N \mapsto_{\mathrm{N}} R$ and $Q \to_{\mathrm{P}}^{*} R$.*

ii) (Backwards simulation) Let M be a λ-term. If $M \mapsto_{\mathrm{N}} Q$, and $Q \to_{\mathrm{P}}^{*} R$, then there exists a λ-term N such that $N \mapsto_{\mathrm{N}} R$ and $M \to_{\mathrm{N}}^{*} N$.

This is a rather complicated solution for a problem caused by, in our opinion, the wrong choice of grammar for CBPV. Since the problem is introduced by (implicit) substitution and arbitrarily allowing for thunking and forcing, our solution is to change exactly how terms are inserted into positions occupied by variables; this will be the approach of Section 4, where we present *essential* CBPV.

3.3 The results for λ-*val* [5]

In this section we will revisit the results shown for a variant of the λ-calculus presented in [10, 5] called the 'simply-typed fine-grained call-by-value λ-calculus' that distinguishes *terms* from *values* through the prefix construct *ret* in [10], called *val* in [5]. It is called CBV throughout [5], but since it differs significantly from the original CBV-λ-calculus, in order to distinguish these we call it the λ-*val*-calculus here. Reduction rules are not presented in [10]; to be able to show our results, we will define them below.

Definition 3.9 (THE λ-*val*-CALCULUS [5]) The terms of the λ-*val*-calculus are defined using the grammar:

$$\begin{array}{rcll} u,v & ::= & x \mid \lambda x.s & (\textit{values}) \\ s,t & ::= & \mathit{val}\,v \mid st & (\textit{expressions}) \end{array}$$

For type assignment, we use the types and variants of the inference rules from Definition 1.2, adding a rule that deals with *val*:

$$(axiom): \frac{}{\Gamma, x{:}A \vdash_v x : A} \qquad (abstr): \frac{\Gamma, x{:}A \vdash_e s : B}{\Gamma \vdash_v \lambda x.s : A \to B}$$

$$(val): \frac{\Gamma \vdash_v v : A}{\Gamma \vdash_e val\, v : A} \qquad (appl): \frac{\Gamma \vdash_e s : A \to B \quad \Gamma \vdash_e t : A}{\Gamma \vdash_e s\, t : B}$$

The name CBV used in [5] for this calculus although it bears little resemblance to CBV-λ: notice that rule (*abstr*) recognises that an abstraction is a value, but it needs to be labelled with the keyword *val* before it can be used in an application. Since it redefines the concept of CBV, there is little claim to be made that the results of [5] deal with the relation between CBN, CBV, and CBPV.

When mapping λ-*val* into CBPV, expressions are interpreted as CBPV computations, so types should be interpreted as CBPV computation types. Since the type syntax for CBPV is different, we need an interpretation of the λ-calculus types into CBPV types.

Definition 3.10 (SIMULATION OF λ-*val* IN CBPV [5]) Terms of the λ-*val*-calculus are translated into those for CBPV through $\overline{\cdot}$:

$$\overline{x} \triangleq x \qquad \overline{val\, v} \triangleq ret\, \overline{v}$$
$$\overline{\lambda x.s} \triangleq \{\lambda x.\overline{s}\} \qquad \overline{s\,t} \triangleq x := \overline{s}; y := \overline{t}; x!y$$

Types are translated using Levy's $\overline{\cdot}$ (Definition 3.4).

Notice that values are only allowed inside applications if preceded by the *val* keyword; this is the main difference between λ-*val* and the traditional CBV λ-calculus, and is (we believe) mainly added to facilitate an interpretation into CBPV. Moreover, the interpretation maps values to values, and expressions to computations.

Forster *et al.* state a type preservation result in [5].Lemma 2.3, which incorrectly restates the result shown in [11] as 'If $\Gamma \vdash_{\mathrm{E}} s : A$, then $\overline{\Gamma} \vdash_{\mathrm{C}} \overline{s} : \overline{A}$ and analogously for values'.[7] We correct and prove the result here for our system.

Lemma 3.11 *If $\Gamma \vdash_v v : A$, then $\overline{\Gamma} \vdash_{\mathrm{V}} \overline{v} : \overline{A}$, and if $\Gamma \vdash_e s : A$, then $\overline{\Gamma} \vdash_{\mathrm{C}} \overline{s} : \mathsf{F}\,\overline{A}$.*

Proof: Simultaneously by induction on the definition of type assignment.

(*axiom*): Then $v \equiv x$, and $\Gamma = \Gamma', x{:}A$; since $\overline{v} \triangleq x$, and $\overline{\Gamma} = \overline{\Gamma'}, x{:}\overline{A}$, by rule (*axiom*) also $\overline{\Gamma} \vdash_{\mathrm{V}} \overline{v} : \overline{A}$.

(*abstr*): Then $A = B \to C$, $v \equiv \lambda x.s$, and $\Gamma, x{:}B \vdash_e s : C$; since $\overline{\Gamma, x{:}B} = \overline{\Gamma}, x{:}\overline{B}$, by induction we have $\overline{\Gamma}, x{:}\overline{B} \vdash_{\mathrm{C}} \overline{s} : \mathsf{F}\,\overline{C}$. We can construct:

$$\dfrac{\dfrac{}{\overline{\Gamma} \vdash_{\mathrm{C}} \lambda x.\overline{s} : \overline{B} \to \mathsf{F}\overline{C}}\,(abstr)}{\overline{\Gamma} \vdash_{\mathrm{V}} \{\lambda x.\overline{s}\} : \mathsf{U}\,(\overline{B} \to \mathsf{F}\overline{C})}\,(thunk)$$

Notice that $\overline{\lambda x.s} \triangleq \{\lambda x.\overline{s}\}$ and $\overline{A} = \mathsf{U}(\overline{B} \to \mathsf{F}\overline{C})$.

(*val*): Then $s \equiv val\, v$, and $\Gamma \vdash_v v : A$; by induction, we have $\overline{\Gamma} \vdash_{\mathrm{V}} \overline{v} : \overline{A}$. Then, by rule (*ret*), $\overline{\Gamma} \vdash_{\mathrm{C}} ret\, \overline{v} : \mathsf{F}\,\overline{A}$.

(*appl*): Then $s \equiv u\, v$, and there exists B such that $\Gamma \vdash_e u : B \to A$ and $\Gamma \vdash_e v : B$. Then, by induction, $\overline{\Gamma} \vdash_{\mathrm{C}} \overline{u} : \mathsf{F}\,\overline{B \to A}$ and $\overline{\Gamma} \vdash_{\mathrm{C}} \overline{v} : \mathsf{F}\,\overline{B}$. Notice that $\overline{B \to A} = \mathsf{U}(\overline{B} \to \mathsf{F}\overline{A})$, and $\overline{u\,v} \triangleq x := \overline{u}; y := \overline{v}; x!y$. We can construct (with $\Gamma' = \overline{\Gamma}, x{:}\mathsf{U}(B \to \mathsf{A}), y{:}B$):

[7] The proofs in Coq provided online for [5] seem to avoid this error.

$$
\begin{array}{ll}
\overline{(val\,\lambda a.(val\,a)\,(val\,a))\,(val\,\lambda a.(val\,a)\,(val\,a))} & \triangleq \\
x := \overline{val\,\lambda a.(val\,a)\,(val\,a)}; y := \overline{val\,\lambda a.(val\,a)\,(val\,a)}; x!y & \triangleq \\
x := ret\,\overline{\lambda a.(val\,a)\,(val\,a)}; y := ret\,\overline{\lambda a.(val\,a)\,(val\,a)}; x!y & \to_{\mathrm{P}} \\
y := ret\,\overline{\lambda a.(val\,a)\,(val\,a)}; \overline{\lambda a.(val\,a)\,(val\,a)}!\,y & \to_{\mathrm{P}} \\
\overline{\lambda a.(val\,a)\,(val\,a)}!\ \overline{\lambda a.(val\,a)\,(val\,a)} & \triangleq \\
\{\lambda a.\overline{(val\,a)\,(val\,a)}\}!\ \overline{\lambda a.(val\,a)\,(val\,a)} & \to_{\mathrm{P}} \\
(\lambda a.\overline{(val\,a)\,(val\,a)})\ \overline{\lambda a.(val\,a)\,(val\,a)} & \triangleq \\
(\lambda a.x := ret\,a; y := ret\,a; x!y)\ \overline{\lambda a.(val\,a)\,(val\,a)} & \to_{\mathrm{P}} \\
x := ret\,\overline{\lambda a.(val\,a)\,(val\,a)}; y := ret\,\overline{\lambda a.(val\,a)\,(val\,a)}; x!y & \triangleq \\
\overline{(val\,\lambda a.(val\,a)\,(val\,a))\,(val\,\lambda a.(val\,a)\,(val\,a))} &
\end{array}
$$

Figure 1. Running $\overline{(val\,\lambda a.(val\,a)\,(val\,a))\,(val\,\lambda a.(val\,a)\,(val\,a))}$ in $\to_{\mathrm{P}}$.

$$
\dfrac{\dfrac{}{\overline{\Gamma}\vdash_{\mathrm{c}} \overline{u} : \mathsf{F}\,\mathsf{U}\,(B\to\mathsf{A})} \quad \dfrac{\dfrac{}{\overline{\Gamma}, x{:}\mathsf{U}\,(B\to\mathsf{A})\vdash_{\mathrm{c}} \overline{v} : \mathsf{F}\,B} \quad \dfrac{\dfrac{\dfrac{}{\Gamma'\vdash_{\mathrm{c}} x : \mathsf{U}\,(B\to\mathsf{F}\,A)}\,(axiom)}{\Gamma'\vdash_{\mathrm{c}} x! : B\to\mathsf{F}\,A}\,(force) \quad \dfrac{}{\Gamma'\vdash_{\mathrm{c}} y : B}\,(axiom)}{\overline{\Gamma}, x{:}\mathsf{U}\,(B\to\mathsf{A}), y{:}B\vdash_{\mathrm{c}} x!y : \mathsf{F}\,A}\,(appl)}{\overline{\Gamma}, x{:}\mathsf{U}\,(B\to\mathsf{A})\vdash_{\mathrm{c}} y := \overline{v}; x!y : \mathsf{F}\,A}\,(assign)}{\overline{\Gamma}\vdash_{\mathrm{c}} x := \overline{u}; y := \overline{v}; x!y : \mathsf{F}\,A}\,(assign)
$$

□

Reduction on *λ-val* is not formally defined in [5], but following common practice, we can assume it to be defined as follows.

Definition 3.12 (Reduction on *λ-val*) *Evaluation contexts* for *λ-val* are defined through:

$$\mathsf{C_V} \ ::= \ [\,] \mid \mathsf{C_V}\,t \mid (val\,v)\,\mathsf{C_V}$$

Reduction $\to_v$ on *λ-val* is (rather awkwardly) defined through:

$$\mathsf{C_V}\,[(val\,\lambda x.s)\,(val\,v)] \ \to_v \ \mathsf{C_V}\,[s\,[v/x]]$$

and $=_v$ is the equivalence relation generated by $\to_v$.

Example 3.13 We consider the reduction of $(val\,\lambda a.(val\,a)\,(val\,a))\,(val\,\lambda a.(val\,a)\,(val\,a))$ (which is the *λ-val* equivalent of the λ-term $(\lambda a.aa)\,(\lambda a.aa)$) which, as we would expect, runs to itself.

$$
\begin{array}{ll}
(val\,\lambda a.(val\,a)\,(val\,a))\,(val\,\lambda a.(val\,a)\,(val\,a)) & \to_v \\
(val\,a)\,(val\,a)\,[\lambda a.(val\,a)\,(val\,a)/a] & = \\
(val\,\lambda a.(val\,a)\,(val\,a))\,(val\,\lambda a.(val\,a)\,(val\,a)) &
\end{array}
$$

The interpretation of this term into CBPV runs as in Figure 1.

As we can see from this example, using the keyword *val* creates a rather cumbersome calculus and notion of reduction, in which it plays no role at all. It seems that the only real reason for using it is to facilitate the encoding results, since it causes the keyword *ret* to be placed inside the interpreted terms. But this was already achieved by Levy's original interpretation (see Definition 3.1).

Regardless of the missing formal definition, the authors claim that their 'translation is correct w.r.t. small-step semantics' [5]. We make the following observation:

Example 3.14 Consider the reduction

$$(val\,\lambda a.val\,a)\,((val\,\lambda b.val\,b)\,(val\,\lambda c.val\,c)) \ \to_v \ (val\,\lambda a.val\,a)\,(val\,\lambda c.val\,c)$$

then under the interpretation we have the reduction given in Figure 2, and

$$
\begin{array}{ll}
\overline{(val\,\lambda a.val\,a)\,((val\,\lambda b.val\,b)\,(val\,\lambda c.val\,c))} & \triangleq \\
x := \overline{val\,\lambda a.val\,a}; y := \overline{(val\,\lambda b.val\,b)\,(val\,\lambda c.val\,c)}; x!y & \triangleq \\
x := ret\,\{\lambda a.ret\,a\}; y := \overline{(val\,\lambda b.val\,b)\,(val\,\lambda c.val\,c)}; x!y & \to_{\mathrm{P}} \\
y := \overline{(val\,\lambda b.val\,b)\,(val\,\lambda c.val\,c)}; \{\lambda a.ret\,a\}!y & \triangleq \\
y := (u := \overline{val\,\lambda b.val\,b}; v := \overline{val\,\lambda c.val\,c}; u!v); \{\lambda a.ret\,a\}!y & \triangleq \\
y := (u := ret\,\{\lambda b.ret\,b\}; v := ret\,\{\lambda c.ret\,c\}; u!v); \{\lambda a.ret\,a\}!y & \to_{\mathrm{P}} \\
y := (v := ret\,\{\lambda c.ret\,c\}; \{\lambda b.ret\,b\}!\,v); \{\lambda a.ret\,a\}!\,y & \to_{\mathrm{P}} \\
y := \{\lambda b.ret\,b\}!\,\{\lambda c.ret\,c\}; \{\lambda a.ret\,a\}!\,y & \to_{\mathrm{P}} \\
y := (\lambda b.ret\,b)\,\{\lambda c.ret\,c\}; \{\lambda a.ret\,a\}!\,y & \to_{\mathrm{P}} \\
y := ret\,\{\lambda c.ret\,c\}; \{\lambda a.ret\,a\}!\,y & \to_{\mathrm{P}} \\
\{\lambda a.ret\,a\}!\,\{\lambda c.ret\,c\} \to_{\mathrm{P}} (\lambda a.ret\,a)\,\{\lambda c.ret\,c\} \to_{\mathrm{P}} ret\,\{\lambda c.ret\,c\} &
\end{array}
$$

Figure 2. Running $\overline{(val\,\lambda a.val\,a)\,((val\,\lambda b.val\,b)\,(val\,\lambda c.val\,c))}$ in $\to_{\mathrm{P}}$.

$$
\begin{array}{lll}
\overline{(val\,\lambda a.val\,a)\,(val\,\lambda c.val\,c)} & \triangleq & x := ret\,\{\lambda a.ret\,a\}; y := ret\,\{\lambda c.ret\,c\}; x!y \to_{\mathrm{P}} \\
y := ret\,\{\lambda c.ret\,c\}; \{\lambda a.ret\,a\}!y & \to_{\mathrm{P}} & \{\lambda a.ret\,a\}!\{\lambda c.ret\,c\} \to_{\mathrm{P}} \\
(\lambda a.ret\,a)\,\{\lambda c.ret\,c\} & \to_{\mathrm{P}} & ret\,\{\lambda c.ret\,c\}
\end{array}
$$

As before, we can only show that if $s \to_v t$, then $\overline{s} =_{\mathrm{P}} \overline{t}$.

Notice that

$$\overline{(val\,\lambda a.val\,a)\,((val\,\lambda b.val\,b)\,(val\,\lambda c.val\,c))} \not\to_{\mathrm{P}} \overline{(val\,\lambda a.val\,a)\,(val\,\lambda c.val\,c)},$$

so single-step reduction is *not* preserved under the interpretation.[8]

As this example suggests, it would be possible to show that the interpretation is correct with respect to large-step semantics. This is of course a weaker property, since only terminating terms can be equated then.

In [5] there is no real motivation given for the departure from the pure λ-calculus with CBV-reduction by adding the keyword *val*. In fact, all results shown in [5] were already claimed in [10], but using (an extension of) the pure λ-calculus with CBV-reduction. In particular, the problem of Theorem 3.3, that reduction is only respected upto equality, is still there.

4 Essential CBPV

In this section we will address a number of the issues we mentioned above by defining *essential* CBPV, a variant of CBPV for which it will be possible to prove stronger results. The main change will be that we will limit the use of thunking and forcing, essentially only allowing for the forcing of variables and thunking of unforced computations. We define a notion of substitution '$M\langle V/x\rangle$' on eCBPV-terms essentially as normal, with the exception of the case $(x!)\langle\{M\}/x\rangle$ which produces M rather than $\{M\}!$, effectively contracting the U-redex that would be created 'on the fly' and thereby making that reduction rule obsolete. This is comparable to dropping the construct *ret* when contracting an F-redex.

Definition 4.1 (eCBPV) *i*) Terms of *essential* CBPV (eCBPV) are defined through the grammar:

$$
\begin{array}{lll}
V, W & ::= & x \mid \{M\} \qquad (M \neq x!) \\
M, N & ::= & x! \mid \lambda x.M \mid MV \mid ret\,V \mid x := M; N
\end{array}
$$

ii) The substitution '$M\langle V/x\rangle$' on eCBPV-terms is defined as follows:

[8] It could of course be that our definition of reduction in Definition 3.12 is not the one the authors of [5] intended, but that seems unlikely.

$$
\begin{array}{rcl@{\qquad}rcl}
x\langle V/x\rangle & = & V & \{M\}\langle V/x\rangle & = & \{M\langle V/x\rangle\} \\
y\langle V/x\rangle & = & y \quad (y \neq x) & (\lambda z.M)\langle V/x\rangle & = & \lambda z.M\langle V/x\rangle \\
(x!)\langle V/x\rangle & = & \begin{cases} z! & (V = z) \\ M & (V = \{M\}) \end{cases} & (ret\, W)\langle V/x\rangle & = & ret\,(W\langle V/x\rangle) \\
(y!)\langle V/x\rangle & = & y! \quad (y \neq x) & (MW)\langle V/x\rangle & = & (M\langle V/x\rangle)\,(W\langle V/x\rangle) \\
 & & & (y := M; N)\langle V/x\rangle & = & y := (M\langle V/x\rangle); (N\langle V/x\rangle)
\end{array}
$$

iii) The basic reduction rules are now defined using this substitution:

$$
\begin{array}{rcl}
(C'): & (\lambda x.M)\,V & \to M\langle V/x\rangle \\
(F'): & x := ret\, V; N & \to N\langle V/x\rangle
\end{array}
$$

iv) eCBPV-reduction is defined through:

$$M \to N \;\Rightarrow\; \mathsf{C}[M] \to \mathsf{C}[N]$$

whenever $M \to N$ through either rule (C') or (F'), where C is a CBPV-evaluation context

$$\mathsf{C} \;::=\; \lceil\rfloor \mid \mathsf{C}V \mid x := \mathsf{C}; M$$

(as in Definition 2.2). We use $\to_{\mathrm{e}}$ for this notion.

Notice the absence of rule (U).

Type assignment for this variant is defined using the rules of Definition 2.4, and subject reduction follows easily, after showing first that our notion of substitution preserves assignable types.

Lemma 4.2 i) If $\Gamma, x{:}B \vdash_{\mathrm{V}} W : A$ *and* $\Gamma \vdash_{\mathrm{V}} V : B$, *then* $\Gamma \vdash_{\mathrm{V}} W\langle V/x\rangle : A$.
ii) If $\Gamma, x{:}B \vdash_{\mathrm{C}} M : \mathsf{A}$ *and* $\Gamma \vdash_{\mathrm{V}} V : B$, *then* $\Gamma \vdash_{\mathrm{C}} M\langle V/x\rangle : \mathsf{A}$.

Proof: By simultaneous induction on the definition of '$\cdot\,\langle V/x\rangle$'. □

Theorem 4.3 *If* $\Gamma \vdash_{\mathrm{C}} M : \mathsf{A}$, *and* $M \to_e^* N$, *then* $\Gamma \vdash_{\mathrm{C}} N : \mathsf{A}$.

Proof: By induction on the definition of reduction; we only show the cases for single-step reduction.

(C'): Then $M = (\lambda x.P)\,V$ and $N = P\langle V/x\rangle$. Then there exists B such that the derivation for $\Gamma \vdash_{\mathrm{C}} M : \mathsf{A}$ is constructed as follows:

$$\dfrac{\dfrac{\boxed{\phantom{\Gamma, x{:}B \vdash_{\mathrm{C}} P : \mathsf{A}}}}{\Gamma \vdash_{\mathrm{C}} \lambda x.P : B \to \mathsf{A}}\,(abstr) \qquad \boxed{\phantom{\Gamma \vdash_{\mathrm{V}} V : B}}}{\Gamma \vdash_{\mathrm{C}} (\lambda x.P)\,V : \mathsf{A}}\,(appl)$$

so in particular $\Gamma, x{:}B \vdash_{\mathrm{C}} P : \mathsf{A}$ and $\Gamma \vdash_{\mathrm{V}} V : B$. Then by Lemma 4.2 we have $\Gamma \vdash_{\mathrm{C}} P\langle V/x\rangle : \mathsf{A}$.

(F'): Then $M = x := ret\, V; P$ and $N = P\langle V/x\rangle$. Then there exists B such that the derivation for $\Gamma \vdash_{\mathrm{C}} M : \mathsf{A}$ is constructed as follows:

$$\dfrac{\dfrac{\boxed{\phantom{\Gamma \vdash_{\mathrm{V}} V : B}}}{\Gamma \vdash_{\mathrm{C}} ret\, V : \mathsf{F}B}\,(ret) \qquad \boxed{\phantom{\Gamma, x{:}B \vdash_{\mathrm{C}} P : \mathsf{A}}}}{\Gamma \vdash_{\mathrm{C}} x := ret\, V; P : \mathsf{A}}\,(assign)$$

so in particular $\Gamma, x{:}B \vdash_{\mathrm{C}} P : \mathsf{A}$ and $\Gamma \vdash_{\mathrm{V}} V : B$. Then by Lemma 4.2 we have $\Gamma \vdash_{\mathrm{C}} P\langle V/x\rangle : \mathsf{A}$.

$(P \to_e Q \Rightarrow M = PV \to_e PV = N)$: Then there exists B such that the derivation for $\Gamma \vdash_{\mathrm{C}} M : \mathsf{A}$ is constructed as follows:

$$\dfrac{\Gamma \vdash_{\mathrm{c}} P : B \to \mathsf{A} \qquad \Gamma \vdash_{\mathrm{v}} V : B}{\Gamma \vdash_{\mathrm{c}} (\lambda x.P)\, V : \mathsf{A}}\ (\mathit{appl})$$

By induction we have $\Gamma \vdash_{\mathrm{c}} P : B \to \mathsf{A}$, and the result follows by rule (appl).

$(P \to_e Q \Rightarrow M = x := P; R \to_e x := Q; R = N)$: We have:

$$\dfrac{\Gamma \vdash_{\mathrm{c}} M : \mathsf{F}B \qquad \Gamma, x{:}B \vdash_{\mathrm{c}} P : \mathsf{A}}{\Gamma \vdash_{\mathrm{c}} x := M; P : \mathsf{A}}\ (\mathit{assign})$$

By induction we have $\Gamma \vdash_{\mathrm{c}} N : \mathsf{F}B$, and the result follows by rule (assign). □

4.1 A CBV-interpretation of Λ in eCBPV

We will now define our CBV interpretation of pure λ-terms into eCBPV and show that it does respect single-step CBV-reduction, and not just equality.

Definition 4.4 The CBV interpretation $\llbracket \cdot \rrbracket^{\lambda}_{\mathrm{v}}$ of λ-terms into eCBPV is defined through:

$$\begin{aligned}
\llbracket x \rrbracket^{\lambda}_{\mathrm{v}} &= \mathit{ret}\, x \\
\llbracket \lambda x.M \rrbracket^{\lambda}_{\mathrm{v}} &= \mathit{ret}\, \{\lambda x. \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}\} \\
\llbracket z N \rrbracket^{\lambda}_{\mathrm{v}} &= y := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; z\,!\,y \\
\llbracket (\lambda z.M)\, N \rrbracket^{\lambda}_{\mathrm{v}} &= y := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; (\lambda z. \llbracket M \rrbracket^{\lambda}_{\mathrm{v}})\,!\,y \\
\llbracket M N \rrbracket^{\lambda}_{\mathrm{v}} &= x := \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; x\,!\,y \quad (M \textit{ not a value})
\end{aligned}$$

So our interpretation deals with all terms basically in the same way as $\llbracket \cdot \rrbracket^{\lambda}_{\mathrm{v}}$ of Definition 3.1, but for the fact that we place the value V in the interpretation of VN *directly*, so without using assignment; this is done in a term rewriting style, so without using (implicit) substitution.

Using this interpretation, we can now show that CBV reduction for the pure λ-calculus is respected by the interpretation. First we show that $\llbracket \cdot \rrbracket^{\lambda}_{\mathrm{v}}$ respects substitution.

Lemma 4.5 (SUBSTITUTION LEMMA FOR $\llbracket \cdot \rrbracket^{\lambda}_{\mathrm{v}}$) *i)* $\llbracket M \rrbracket^{\lambda}_{\mathrm{v}} \langle w/z \rangle = \llbracket M[w/z] \rrbracket^{\lambda}_{\mathrm{v}}$.
ii) $\llbracket M \rrbracket^{\lambda}_{\mathrm{v}} \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle = \llbracket M[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}$.

Proof: i) By straightforward induction on the definition of substitution.

ii) $(M \equiv z)$: $\llbracket z \rrbracket^{\lambda}_{\mathrm{v}} \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle \triangleq (\mathit{ret}\, z) \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle \triangleq \mathit{ret}\, \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\} \triangleq \llbracket \lambda w.R \rrbracket^{\lambda}_{\mathrm{v}} \triangleq \llbracket z[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}$

$(M \equiv u \textit{ and } u \neq z)$: $\llbracket u \rrbracket^{\lambda}_{\mathrm{v}} \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle \triangleq (\mathit{ret}\, u) \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle \triangleq \mathit{ret}\, u \triangleq \llbracket u \rrbracket^{\lambda}_{\mathrm{v}} \triangleq \llbracket u[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}$

$(M \equiv \lambda u.N)$: $\llbracket \lambda u.N \rrbracket^{\lambda}_{\mathrm{v}} \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle \triangleq (\mathit{ret}\, \{\lambda u. \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}\}) \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle \triangleq \mathit{ret}\, \{\lambda u. (\llbracket N \rrbracket^{\lambda}_{\mathrm{v}} \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle)\} = (IH)\ \mathit{ret}\, \{\lambda u. \llbracket N[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}\} \triangleq \llbracket \lambda u.N[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}$

$(M \equiv xQ)$: $\llbracket xQ \rrbracket^{\lambda}_{\mathrm{v}} \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle \triangleq (y := \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}}; x\,!\,y) \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle \triangleq y := \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}} \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle; x\,!\,y = (IH)\ y := \llbracket Q[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}; x\,!\,y \triangleq \llbracket xQ[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}} \triangleq \llbracket (xQ)[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}$

$(M \equiv (\lambda x.P)\, Q)$: $\llbracket (\lambda x.P)\, Q \rrbracket^{\lambda}_{\mathrm{v}} \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle \triangleq (y := \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}}; \llbracket \lambda x.P \rrbracket^{\lambda}_{\mathrm{v}}\,!\,y) \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle \triangleq y := (\llbracket Q \rrbracket^{\lambda}_{\mathrm{v}} \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle); (\llbracket \lambda x.P \rrbracket^{\lambda}_{\mathrm{v}} \langle \{\lambda w. \llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z \rangle)\,!\,y = (IH)\ y := \llbracket Q[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}; \llbracket (\lambda x.P)[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}\,!\,y \triangleq \llbracket ((\lambda x.P)[\lambda w.R/z])\, Q[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}} \triangleq \llbracket ((\lambda x.P)\, Q)[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}$

$$
\begin{array}{ll}
(M \equiv PQ,\ P \text{ not a } \lambda\text{-value}):\ \ \llbracket PQ \rrbracket^{\lambda}_{\mathrm{v}} \langle \{\lambda w.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z\rangle & \triangleq \\
\quad (x := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}}; x!y)\langle \{\lambda w.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z\rangle & \triangleq \\
\quad (x := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; y := \langle \{\lambda w.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z\rangle; x!y \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}} \langle \{\lambda w.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/z\rangle) & = (IH) \\
\quad x := \llbracket P[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket Q[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}}; x!y & \triangleq \\
\quad \llbracket P[\lambda w.R/z] Q[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}} & = \\
\quad \llbracket (PQ)[\lambda w.R/z] \rrbracket^{\lambda}_{\mathrm{v}} & \square
\end{array}
$$

Using this lemma, we can now show:

Theorem 4.6 *If $M \to_{\mathrm{v}} N$, then $\llbracket M \rrbracket^{\lambda}_{\mathrm{v}} \to^{*}_{e} \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}$.*

Proof: $((\lambda x.M)z \to_{\mathrm{v}} M[z/x])$: $\llbracket (\lambda x.M) z \rrbracket^{\lambda}_{\mathrm{v}} \triangleq y := ret\, z; (\lambda x.\llbracket M \rrbracket^{\lambda}_{\mathrm{v}}) y \to_{\mathrm{p}} (\lambda x.\llbracket M \rrbracket^{\lambda}_{\mathrm{v}}) z \to_{\mathrm{p}} \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}\langle z/x\rangle = (4.5)\ \llbracket M[z/x] \rrbracket^{\lambda}_{\mathrm{v}}$

$((\lambda x.M)(\lambda z.R) \to_{\mathrm{v}} M[\lambda z.R/x])$: $\llbracket (\lambda x.M)(\lambda z.R) \rrbracket^{\lambda}_{\mathrm{v}} \triangleq y := \llbracket \lambda z.R \rrbracket^{\lambda}_{\mathrm{v}}; (\lambda x.\llbracket M \rrbracket^{\lambda}_{\mathrm{v}})!y \triangleq y := ret\{\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}; (\lambda x.\llbracket M \rrbracket^{\lambda}_{\mathrm{v}})!y \to_{\mathrm{p}} (\lambda x.\llbracket M \rrbracket^{\lambda}_{\mathrm{v}})\{\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\} \to_{\mathrm{p}} \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}\langle \{\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/x\rangle = (4.5)\ \llbracket M[\lambda z.R/x] \rrbracket^{\lambda}_{\mathrm{v}}$

$(M \to_{\mathrm{v}} z \Rightarrow MP \to_{\mathrm{v}} zP)$: $\llbracket MP \rrbracket^{\lambda}_{\mathrm{v}} \triangleq x := \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; x!y \to^{*}_{\mathrm{p}} (IH)$
$x := \llbracket z \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; x!y \triangleq x := ret\, z; y := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; x!y \to^{*}_{\mathrm{p}}$
$y := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; z!y \triangleq \llbracket zP \rrbracket^{\lambda}_{\mathrm{v}}$

$(M \to_{\mathrm{v}} \lambda z.N \Rightarrow MP \to_{\mathrm{v}} (\lambda z.N)P)$: $\llbracket MP \rrbracket^{\lambda}_{\mathrm{v}} \triangleq x := \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; x!y \to^{*}_{\mathrm{p}} (IH)$
$x := \llbracket \lambda z.N \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; x!y \triangleq x := ret\{\lambda z.\llbracket N \rrbracket^{\lambda}_{\mathrm{v}}\}; y := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; x!y \to^{*}_{\mathrm{p}}$
$y := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; (\lambda z.\llbracket N \rrbracket^{\lambda}_{\mathrm{v}}) y \triangleq \llbracket (\lambda z.N) P \rrbracket^{\lambda}_{\mathrm{v}}$

$(M \to_{\mathrm{v}} N \Rightarrow MP \to_{\mathrm{v}} NP,\ N \text{ not a value})$: $\llbracket MP \rrbracket^{\lambda}_{\mathrm{v}} \triangleq x := \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; x!y \to^{*}_{\mathrm{p}} (IH)$
$x := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; x!y \triangleq \llbracket NP \rrbracket^{\lambda}_{\mathrm{v}}$

$(M \to_{\mathrm{v}} N \Rightarrow zM \to^{*}_{\mathrm{v}} zN)$: $\llbracket zM \rrbracket^{\lambda}_{\mathrm{v}} \triangleq y := \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}; z!y \to^{*}_{\mathrm{p}} (IH)\ \ y := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; z!y \triangleq \llbracket zN \rrbracket^{\lambda}_{\mathrm{v}}$

$(M \to_{\mathrm{v}} N \Rightarrow (\lambda z.R)M \to^{*}_{\mathrm{v}} (\lambda z.R)N)$: $\llbracket (\lambda z.R) M \rrbracket^{\lambda}_{\mathrm{v}} \triangleq y := \llbracket M \rrbracket^{\lambda}_{\mathrm{v}}; (\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}) y \to^{*}_{\mathrm{p}} (IH)$
$y := \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}; (\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}) y \triangleq \llbracket (\lambda z.R) N \rrbracket^{\lambda}_{\mathrm{v}}$ $\square$

Notice that this result was shown for reduction, in contrast to Theorem 3.3 which was shown for equality.

We can also show that type assignment for the pure λ-calculus is respected by the interpretation $\llbracket \cdot \rrbracket^{\lambda}_{\mathrm{v}}$.

Lemma 4.7 *If $\Gamma \vdash M : A$, then $\overline{\Gamma} \vdash_{\mathrm{C}} \llbracket M \rrbracket^{\lambda}_{\mathrm{v}} : \mathsf{F}\,\overline{A}$.*

Proof: By induction on definition of type assignment.

(Ax): Then $M \equiv x$, and $\Gamma = \Gamma', x{:}A$, and $\overline{\Gamma} \triangleq \overline{\Gamma'}, x{:}\overline{A}$. We can construct:

$$\dfrac{\dfrac{}{\overline{\Gamma} \vdash_{\mathrm{v}} x : \overline{A}}\ (axiom)}{\overline{\Gamma} \vdash_{\mathrm{c}} ret\, x : \mathsf{F}\,\overline{A}}\ (ret)$$

Notice that $\overline{x} \triangleq ret\, x$.

$(\to I)$: Then $A = B \to C$, $M \equiv \lambda x.N$, and $\Gamma, x{:}B \vdash_{\mathrm{E}} N : C$; since $\overline{\Gamma, x{:}B} \triangleq \overline{\Gamma}, x{:}\overline{B}$, by induction we have $\overline{\Gamma}, x{:}\overline{B} \vdash_{\mathrm{C}} \llbracket N \rrbracket^{\lambda}_{\mathrm{v}} : \mathsf{F}\,\overline{C}$. We can construct:

$$\dfrac{\dfrac{\dfrac{\dfrac{\boxed{\phantom{\overline{\Gamma}, x{:}\overline{B} \vdash_{\mathrm{c}} \llbracket N \rrbracket^{\lambda}_{\mathrm{v}} : \mathsf{F}\,\overline{C}}}}}{\overline{\Gamma}, x{:}\overline{B} \vdash_{\mathrm{c}} \llbracket N \rrbracket^{\lambda}_{\mathrm{v}} : \mathsf{F}\,\overline{C}}}{\overline{\Gamma} \vdash_{\mathrm{c}} \lambda x.\llbracket N \rrbracket^{\lambda}_{\mathrm{v}} : \overline{B} \to \mathsf{F}\,\overline{C}}\ (abstr)}{\overline{\Gamma} \vdash_{\mathrm{v}} \{\lambda x.\llbracket N \rrbracket^{\lambda}_{\mathrm{v}}\} : \mathsf{U}\,(\overline{B} \to \mathsf{F}\,\overline{C})}\ (thunk)}{\overline{\Gamma} \vdash_{\mathrm{c}} ret\,\{\lambda x.\llbracket N \rrbracket^{\lambda}_{\mathrm{v}}\} : \mathsf{F}\,\mathsf{U}\,(\overline{B} \to \mathsf{F}\,\overline{C})}\ (ret)$$

Notice that $\llbracket \lambda x.N \rrbracket^{\lambda}_{\mathrm{v}} \triangleq \mathit{ret}\,\{\lambda x.\llbracket N \rrbracket^{\lambda}_{\mathrm{v}}\}$ and $\overline{A} \triangleq \mathsf{U}(\overline{B}{\to}\mathsf{F}\overline{C})$.

$(\to E)$: Then $M \equiv PQ$, and there exists B such that $\Gamma \vdash_{\mathrm{E}} P : B \to A$ and $\Gamma \vdash_{\mathrm{E}} Q : B$. Then, by induction, $\overline{\Gamma} \vdash_{\mathrm{C}} \llbracket P \rrbracket^{\lambda}_{\mathrm{v}} : \mathsf{F}\,\overline{B{\to}A}$ and $\overline{\Gamma} \vdash_{\mathrm{C}} \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}} : \mathsf{F}\,\overline{B}$. Notice that $\overline{B{\to}A} \triangleq \mathsf{U}(\overline{B}{\to}\mathsf{F}\overline{A})$. We have three cases to consider:

$(P \equiv z)$: From $\overline{\Gamma} \vdash_{\mathrm{C}} \llbracket z \rrbracket^{\lambda}_{\mathrm{v}} : \mathsf{F}\,\overline{B{\to}A} \triangleq \overline{\Gamma} \vdash_{\mathrm{C}} \mathit{ret}\,z : \mathsf{F}\,\mathsf{U}(\overline{B}{\to}\mathsf{F}\overline{A})$ we know that $z : \mathsf{U}(\overline{B}{\to}\mathsf{F}\overline{A}) \in \overline{\Gamma}$, so we can construct:

$$\dfrac{\overline{\Gamma} \vdash_{\mathrm{C}} \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}} : \mathsf{F}\,\overline{B} \qquad \dfrac{\dfrac{\overline{\Gamma, y{:}\overline{B} \vdash_{\mathrm{C}} z : \mathsf{U}\,(\overline{B}{\to}\mathsf{F}\overline{A})}\,(\mathit{axiom})}{\overline{\Gamma, y{:}\overline{B} \vdash_{\mathrm{C}} z! : \overline{B}{\to}\mathsf{F}\overline{A}}\,(\mathit{force})} \qquad \dfrac{}{\overline{\Gamma, y{:}\overline{B} \vdash_{\mathrm{C}} y : \overline{B}}\,(\mathit{axiom})}}{\overline{\Gamma, y{:}\overline{B} \vdash_{\mathrm{C}} (z!)\,y : \mathsf{F}\overline{A}}\,(\mathit{appl})}}{\overline{\Gamma} \vdash_{\mathrm{C}} y := \overline{Q}; z!y : \mathsf{F}\overline{A}}\,(\mathit{assign})$$

Notice that $\llbracket zQ \rrbracket^{\lambda}_{\mathrm{v}} \triangleq y := \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}}; z!y$.

$(P \equiv \lambda z.R)$: From $\overline{\Gamma} \vdash_{\mathrm{C}} \llbracket \lambda z.R \rrbracket^{\lambda}_{\mathrm{v}} : \mathsf{F}\,\overline{B{\to}A} \triangleq \overline{\Gamma} \vdash_{\mathrm{C}} \mathit{ret}\,\{\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\} : \mathsf{F}\,\mathsf{U}(\overline{B}{\to}\mathsf{F}\overline{A})$, we know that in a subderivation $\overline{\Gamma} \vdash_{\mathrm{C}} \lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}} : \overline{B}{\to}\mathsf{F}\overline{A}$ is shown, with which we can construct:

$$\dfrac{\overline{\Gamma} \vdash_{\mathrm{C}} \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}} : \mathsf{F}\,\overline{B} \qquad \dfrac{\overline{\Gamma, y{:}\overline{B} \vdash_{\mathrm{C}} \lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}} : \overline{B}{\to}\mathsf{F}\overline{A}} \qquad \dfrac{}{\overline{\Gamma, y{:}\overline{B} \vdash_{\mathrm{C}} y : \overline{B}}\,(\mathit{axiom})}}{\overline{\Gamma, y{:}\overline{B} \vdash_{\mathrm{C}} \{\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}!y : \mathsf{F}\overline{A}}\,(\mathit{appl})}}{\overline{\Gamma} \vdash_{\mathrm{C}} y := \overline{Q}; (\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}})!y : \mathsf{F}\overline{A}}\,(\mathit{assign})$$

Notice that $\llbracket (\lambda z.R)\,Q \rrbracket^{\lambda}_{\mathrm{v}} \triangleq y := \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}}; \{\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}!y$.

$(\mathit{Otherwise})$: We can construct (with $\Gamma' = \overline{\Gamma}, x{:}\mathsf{U}(\overline{B}{\to}\mathsf{F}\overline{A}), y{:}\overline{B}$):

$$\dfrac{\overline{\Gamma} \vdash_{\mathrm{C}} \llbracket P \rrbracket^{\lambda}_{\mathrm{v}} : \mathsf{F}\,\mathsf{U}(\overline{B}{\to}\mathsf{F}\overline{A}) \qquad \dfrac{\overline{\Gamma} \vdash_{\mathrm{C}} \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}} : \mathsf{F}\,\overline{B} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma' \vdash_{\mathrm{C}} x : \mathsf{U}\,(\overline{B}{\to}\mathsf{F}\overline{A})}\,(\mathit{axiom})}{\Gamma' \vdash_{\mathrm{C}} x! : \overline{B}{\to}\mathsf{F}\overline{A}}\,(\mathit{force}) \qquad \dfrac{}{\Gamma' \vdash_{\mathrm{C}} y : \overline{B}}\,(\mathit{axiom})}{\overline{\Gamma}, x{:}\mathsf{U}\,(\overline{B}{\to}\mathsf{F}\overline{A}), y{:}\overline{B} \vdash_{\mathrm{C}} (x!)\,y : \mathsf{F}\overline{A}}\,(\mathit{appl})}{\overline{\Gamma}, x{:}\mathsf{U}\,(\overline{B}{\to}\mathsf{F}\overline{A}) \vdash_{\mathrm{C}} y := \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}}; (x!)\,y : \mathsf{F}\overline{A}}\,(\mathit{assign})}{\overline{\Gamma} \vdash_{\mathrm{C}} x := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}}; x!y : \mathsf{F}\overline{A}}\,(\mathit{assign})$$

Notice that $\llbracket PQ \rrbracket^{\lambda}_{\mathrm{v}} \triangleq x := \llbracket P \rrbracket^{\lambda}_{\mathrm{v}}; y := \llbracket Q \rrbracket^{\lambda}_{\mathrm{v}}; x!y$. □

4.2 A CBN-interpretation of Λ in eCBPV

We can achieve a stronger result for CBN-reduction as well, as we will show now. First we modify the interpretation of Definition 3.6.

Definition 4.8 The CBN interpretation $\llbracket \cdot \rrbracket^{\lambda}_{\mathrm{N}}$ of λ-terms into eCBPV is defined through:

$$\begin{aligned}
\llbracket x \rrbracket^{\lambda}_{\mathrm{N}} &= x! \\
\llbracket \lambda x.M \rrbracket^{\lambda}_{\mathrm{N}} &= \lambda x.\llbracket M \rrbracket^{\lambda}_{\mathrm{N}} \\
\llbracket Mx \rrbracket^{\lambda}_{\mathrm{N}} &= \llbracket M \rrbracket^{\lambda}_{\mathrm{N}}\,x \\
\llbracket MN \rrbracket^{\lambda}_{\mathrm{N}} &= \llbracket M \rrbracket^{\lambda}_{\mathrm{N}}\{\llbracket N \rrbracket^{\lambda}_{\mathrm{N}}\} \quad (N \textit{ not a variable})
\end{aligned}$$

Notice that this corresponds to Levy's CBN-interpretation in Definition 3.6, but for the application case, where we avoid to define $\llbracket Mx \rrbracket^{\lambda}_{\mathrm{N}} = \llbracket M \rrbracket^{\lambda}_{\mathrm{N}}\{x!\}$, but directly write $\llbracket M \rrbracket^{\lambda}_{\mathrm{N}}x$.

Now, other than in Example 3.7 we get:

$$\begin{aligned}
\llbracket (\lambda x.xx)\,(\lambda x.xx) \rrbracket^{\lambda}_{\mathrm{N}} &\triangleq (\lambda x.x!x)\{\lambda x.x!x\} \\
&\to_{\mathrm{e}} (x!x)\,\langle\{\lambda x.x!x\}/x\rangle \\
&= x!\,\langle\{\lambda x.x!x\}/x\rangle\ x\,\langle\{\lambda x.x!x\}/x\rangle \\
&= (\lambda x.x!x)\{\lambda x.x!x\}
\end{aligned}$$

This interpretation respects term substitution:

Lemma 4.9 (Substitution lemma for $\llbracket\cdot\rrbracket^{\lambda}_{\mathrm{N}}$) *i)* $\llbracket M\rrbracket^{\lambda}_{\mathrm{N}}\langle w/z\rangle = \llbracket M[w/z]\rrbracket^{\lambda}_{\mathrm{N}}$.
ii) $\llbracket M\rrbracket^{\lambda}_{\mathrm{N}}\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/x\rangle = \llbracket M[N/x]\rrbracket^{\lambda}_{\mathrm{N}}$, *if N is not a variable.*

Proof: i) By straightforward induction on the definition of $\cdot\langle\cdot/\cdot\rangle$.

ii) $(M\equiv z)$: $\llbracket z\rrbracket^{\lambda}_{\mathrm{N}}\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle = z!\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle = \llbracket N\rrbracket^{\lambda}_{\mathrm{N}} = \llbracket z[N/z]\rrbracket^{\lambda}_{\mathrm{N}}$

$(M\equiv u \text{ and } u\neq z)$: $\llbracket u\rrbracket^{\lambda}_{\mathrm{N}}\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle = (u!)\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle = u! = \llbracket u\rrbracket^{\lambda}_{\mathrm{N}} = \llbracket u[N/z]\rrbracket^{\lambda}_{\mathrm{N}}$

$(M\equiv \lambda u.R)$: $\llbracket\lambda u.R\rrbracket^{\lambda}_{\mathrm{N}}\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle = (\lambda u.\llbracket R\rrbracket^{\lambda}_{\mathrm{N}})\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle = \lambda u.(\llbracket R\rrbracket^{\lambda}_{\mathrm{N}}\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle)$
$= (IH)\quad \lambda u.\llbracket R[N/z]\rrbracket^{\lambda}_{\mathrm{N}} = \llbracket\lambda u.R[N/z]\rrbracket^{\lambda}_{\mathrm{N}}$

$(M\equiv Pz)$: $\llbracket Pz\rrbracket^{\lambda}_{\mathrm{N}}\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle = (\llbracket P\rrbracket^{\lambda}_{\mathrm{N}}z)\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle = \llbracket P\rrbracket\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\} = (IH)$
$\llbracket P[N/z]\rrbracket^{\lambda}_{\mathrm{N}}\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\} = \llbracket P[N/z]N\rrbracket^{\lambda}_{\mathrm{N}} = \llbracket(Pz)[N/z]\rrbracket^{\lambda}_{\mathrm{N}}$

$(M\equiv Px,\ x\neq z)$: $\llbracket Px\rrbracket^{\lambda}_{\mathrm{N}}\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle = (\llbracket P\rrbracket^{\lambda}_{\mathrm{N}}x)\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle = \llbracket P\rrbracket\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle x = (IH)$
$\llbracket P[N/z]\rrbracket^{\lambda}_{\mathrm{N}}x = \llbracket P[N/z]x\rrbracket^{\lambda}_{\mathrm{N}} = \llbracket(Px)[N/z]\rrbracket^{\lambda}_{\mathrm{N}}$

$(M\equiv PQ)$: $\llbracket PQ\rrbracket^{\lambda}_{\mathrm{N}}\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle = (\llbracket P\rrbracket^{\lambda}_{\mathrm{N}}\{\llbracket Q\rrbracket^{\lambda}_{\mathrm{N}}\})\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle =$
$\llbracket P\rrbracket^{\lambda}_{\mathrm{N}}\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle\{\llbracket Q\rrbracket^{\lambda}_{\mathrm{N}}\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rangle\} = (IH)\ \llbracket P[N/z]\rrbracket^{\lambda}_{\mathrm{N}}\{\llbracket Q[N/z]\rrbracket^{\lambda}_{\mathrm{N}}\} =$
$\llbracket P[N/z]Q[N/z]\rrbracket^{\lambda}_{\mathrm{N}} = \llbracket(PQ)[N/z]\rrbracket^{\lambda}_{\mathrm{N}}$ □

With this result we can now show that the interpretation respects CBN-reduction.

Theorem 4.10 *If $M\to_{\mathrm{N}} N$, then $\llbracket M\rrbracket^{\lambda}_{\mathrm{N}}\to_e^*\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}$.*

Proof: $((\lambda x.M)y\to_{\mathrm{N}} M[y/x])$: $\llbracket(\lambda x.M)y\rrbracket^{\lambda}_{\mathrm{N}} \triangleq (\lambda x.\llbracket M\rrbracket^{\lambda}_{\mathrm{N}})y \to_{\mathrm{e}} \llbracket M\rrbracket^{\lambda}_{\mathrm{N}}\langle y/x\rangle = (4.9)$
$\llbracket M[y/x]\rrbracket^{\lambda}_{\mathrm{N}}$

$((\lambda x.M)N\to_{\mathrm{N}} M[N/x]$, *$N$ not a variable*$)$: $\llbracket(\lambda x.M)N\rrbracket^{\lambda}_{\mathrm{N}} \triangleq \lambda x.\llbracket M\rrbracket^{\lambda}_{\mathrm{N}}\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\} \to_{\mathrm{e}}$
$\llbracket M\rrbracket^{\lambda}_{\mathrm{N}}\langle\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/x\rangle = (4.9)\ \llbracket M[N/x]\rrbracket^{\lambda}_{\mathrm{N}}$

$(M\to_{\mathrm{N}} N\Rightarrow Mx\to_{\mathrm{N}} Nx)$: $\llbracket Mx\rrbracket^{\lambda}_{\mathrm{N}} \triangleq \llbracket M\rrbracket^{\lambda}_{\mathrm{N}}x \to_{\mathrm{e}}^*(IH)\ \llbracket N\rrbracket^{\lambda}_{\mathrm{N}}x \triangleq \llbracket Nx\rrbracket^{\lambda}_{\mathrm{N}}$

$(M\to_{\mathrm{N}} N\Rightarrow MP\to_{\mathrm{N}} NP$, *with P not a variable*$)$: $\llbracket MP\rrbracket^{\lambda}_{\mathrm{N}} \triangleq \llbracket M\rrbracket^{\lambda}_{\mathrm{N}}\{\llbracket P\rrbracket^{\lambda}_{\mathrm{N}}\} \to_{\mathrm{e}}^*(IH)$
$\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\{\llbracket P\rrbracket^{\lambda}_{\mathrm{N}}\} \triangleq \llbracket NP\rrbracket^{\lambda}_{\mathrm{N}}$ □

For the preservation of type assignment under $\llbracket\cdot\rrbracket^{\lambda}_{\mathrm{N}}$, we need first to map the types for Curry's system to those for CBPV, in a way befitting the interpretation; for this we can use Levy's CBN-type interpretation directly.

Definition 4.11 (Simple type interpretation [10]) The type interpretation '$\underline{\cdot}$' is defined as:

$$\begin{aligned}\underline{\varphi} &= \mathsf{F}\varphi\\ \underline{A\to B} &= \mathsf{U}\underline{A}\to\underline{B}\end{aligned}$$

and the environment interpretation as:

$$\underline{\Gamma} = \{\,x{:}\mathsf{U}\underline{A} \mid x{:}A\in\Gamma\,\}$$

We can now show that the CBN-interpretation preserves type assignment.

Theorem 4.12 *If $\Gamma\vdash M:A$, then $\underline{\Gamma}\vdash_{\mathrm{C}}\llbracket M\rrbracket^{\lambda}_{\mathrm{N}}:\underline{A}$.*

Proof: By induction on the structure of derivations.

(Ax): Then $M\equiv x$, and $x{:}A\in\Gamma$. Then $x{:}U\underline{A}\in\underline{\Gamma}$ and we can derive:

$$\dfrac{\dfrac{}{\underline{\Gamma}\vdash_{\mathrm{V}} x:\mathsf{U}\underline{A}}\,(axiom)}{\underline{\Gamma}\vdash_{\mathrm{C}} x!:\mathsf{F}\,\mathsf{U}\underline{A}}\,(force)$$

and $\llbracket x\rrbracket^{\lambda}_{\mathrm{N}} \triangleq x!$.

$(\to I)$: Then $M \equiv \lambda x.N$, $A = B \to C$, and $\Gamma, x{:}B \vdash N : C$. By induction, we get $\underline{\Gamma}, x{:}\mathsf{U}\underline{B} \vdash_{\mathrm{C}} \llbracket N \rrbracket^{\lambda}_{\mathbf{N}} : \underline{C}$, and we can construct:

$$\dfrac{\boxed{\phantom{\underline{\Gamma}, x{:}\mathsf{U}\underline{B} \vdash_{\mathrm{C}} \llbracket N \rrbracket : \underline{C}}}}{\underline{\Gamma} \vdash_{\mathrm{C}} \lambda x.\llbracket N \rrbracket : \mathsf{U}\underline{B} \to \underline{C}}\ (abstr)$$

and $\llbracket \lambda x.N \rrbracket^{\lambda}_{\mathbf{N}} \triangleq \lambda x.\llbracket N \rrbracket$ and $\underline{B \to C} = \mathsf{U}\underline{B} \to \underline{C}$.

$(\to E)$: Then $M \equiv PQ$, and there exists B such that $\Gamma \vdash P : B \to A$ and $\Gamma \vdash Q : B$. By induction, we get $\underline{\Gamma} \vdash_{\mathrm{C}} \llbracket P \rrbracket : \underline{B \to A}$; also, $\underline{B \to A} = \mathsf{U}\underline{B} \to \underline{A}$. We have two cases:

$(Q \equiv x)$: Then $x{:}B \in \Gamma$ and $x{:}U\underline{B} \in \underline{\Gamma}$, and we can construct:

$$\dfrac{\boxed{\phantom{\underline{\Gamma} \vdash_{\mathrm{C}} \llbracket P \rrbracket^{\lambda}_{\mathbf{N}} : \mathsf{U}\underline{B} \to \underline{A}}} \qquad \dfrac{}{\underline{\Gamma} \vdash_{\mathrm{V}} x : \mathsf{U}\underline{B}}\ (Ax)}{\underline{\Gamma} \vdash_{\mathrm{C}} \llbracket P \rrbracket^{\lambda}_{\mathbf{N}} x : \underline{A}}\ (appl)$$

and $\llbracket Px \rrbracket^{\lambda}_{\mathbf{N}} \triangleq \llbracket P \rrbracket^{\lambda}_{\mathbf{N}} x$.

$(Q \neq x)$: By induction, we have $\underline{\Gamma} \vdash_{\mathrm{C}} \llbracket Q \rrbracket : \underline{B}$, and can construct:

$$\dfrac{\boxed{\phantom{\underline{\Gamma} \vdash_{\mathrm{C}} \llbracket P \rrbracket^{\lambda}_{\mathbf{N}} : \mathsf{U}\underline{B} \to \underline{A}}} \qquad \dfrac{\boxed{\phantom{\underline{\Gamma} \vdash_{\mathrm{C}} \llbracket Q \rrbracket^{\lambda}_{\mathbf{N}} : \underline{B}}}}{\underline{\Gamma} \vdash_{\mathrm{V}} \{\llbracket Q \rrbracket^{\lambda}_{\mathbf{N}}\} : \mathsf{U}\underline{B}}\ (thunk)}{\underline{\Gamma} \vdash_{\mathrm{C}} \llbracket P \rrbracket^{\lambda}_{\mathbf{N}} \{\llbracket Q \rrbracket^{\lambda}_{\mathbf{N}}\} : \underline{A}}\ (appl)$$

and $\llbracket PQ \rrbracket^{\lambda}_{\mathbf{N}} \triangleq \llbracket P \rrbracket^{\lambda}_{\mathbf{N}} \{\llbracket Q \rrbracket^{\lambda}_{\mathbf{N}}\}$. □

5 The calculus $\overline{\lambda}\mu\tilde{\mu}$

We will now give a short summary of Curien and Herbelin's calculus $\overline{\lambda}\mu\tilde{\mu}$, as first presented in [3]. In its typed version, $\overline{\lambda}\mu\tilde{\mu}$ is a proof-term syntax for a classical sequent calculus that treats a logic with focus, and can be seen as an extension of Parigot's $\lambda\mu$ and a variant of Gentzen's LK. As in $\lambda\mu$, for $\overline{\lambda}\mu\tilde{\mu}$ there are two sets of variables: x, y, z, *etc.*, label the types of the hypotheses and α, β, γ, *etc.*, label the types of the conclusions. The syntax of $\overline{\lambda}\mu\tilde{\mu}$ has three different categories: commands, terms, and contexts (or environments). Commands c form the computational units in $\overline{\lambda}\mu\tilde{\mu}$ and are composed of a pair $\langle v \mid e \rangle$ of a term v and its context (or environment) e that can interact.

Reduction in $\overline{\lambda}\mu\tilde{\mu}$ is dual, in that both parameter call and context call are represented: parameter call through the context $\tilde{\mu}x.c$ that can pull the corresponding term in to the places marked by x, and context call through the term $\mu\alpha.c$ that places the corresponding context in the places marked by α.

Correspondingly, these constructs are typed by three kinds of sequents: the usual sequents $\Gamma \vdash \Delta$ type commands, while the sequents typing terms (resp. contexts) are of the form $\Gamma \vdash A \mid \Delta$ (resp. $\Gamma \mid A \vdash \Delta$), marking the conclusion (resp. hypothesis) A as *active*.

Definition 5.1 (Commands, Terms, and Contexts [3]) There are three categories of expressions in $\overline{\lambda}\mu\tilde{\mu}$, defined by:

$$\begin{array}{lll} c & ::= & \langle v \mid e \rangle & (commands) \\ v & ::= & x \mid \lambda x.v \mid \mu\beta.c & (terms) \\ e & ::= & \alpha \mid v{\cdot}e \mid \tilde{\mu}x.c & (contexts) \end{array}$$

Here λ, μ, and $\tilde{\mu}$ are binders, and the notion of free or bound term and context variables is defined as usual.

With conventional notations about contexts (*i.e.* seeing contexts as terms with a hole), $v \cdot e$

can be thought of as $e[[\,]\,v]$, and the context $v_1\cdot(\cdots(v_n\cdot\alpha)\cdots)$ (we can omit these brackets and write $v_1\cdot\cdots\cdot v_n\cdot\alpha$) as a *stack* (see Example 5.8); $\mu\alpha.c$ is inherited from $\lambda\mu$, as is $\langle v \mid \alpha\rangle$ which corresponds to $\lambda\mu$'s *naming* construct $[\alpha]v$, giving name α to the implicit output name of v; the construct $\tilde{\mu}x.c$ can be thought of as $\mathsf{let}\ x = [\,]\ \mathsf{in}\ c$.

Commands can be computed (thus eliminating the cut in the corresponding proof):

Definition 5.2 (Reduction in $\overline{\lambda}\mu\tilde{\mu}$ [3, 8]) Let $c\,[e/\beta]$ stand for the implicit substitution of the free occurrences of the context variable β by the context e, and $c\,[v/x]$ for that of x by the term v. The reduction rules are defined by:

logical rules *extensional rules*

$$(\lambda):\ \langle \lambda x.v_1 \mid v_2\cdot e\rangle \to \langle v_2 \mid \tilde{\mu}x.\langle v_1 \mid e\rangle\rangle \qquad (\eta):\ \lambda x.\mu\beta.\langle v \mid x\cdot\beta\rangle \to v \quad (x,\beta\notin fv(v))$$

$$(\mu):\ \langle \mu\beta.c \mid e\rangle \to c\,[e/\beta] \qquad (\eta\mu):\ \mu\alpha.\langle v \mid \alpha\rangle \to v \quad (\alpha\notin fv(v))$$

$$(\tilde{\mu}):\ \langle v \mid \tilde{\mu}x.c\rangle \to c\,[v/x] \qquad (\eta\tilde{\mu}):\ \tilde{\mu}x.\langle x \mid e\rangle \to e \quad (x\notin fv(e))$$

contextual rules

$$v\to v' \Rightarrow \begin{cases} \langle v \mid e\rangle \to \langle v' \mid e\rangle \\ \lambda x.v \to \lambda x.v' \\ v\cdot e \to v'\cdot e \end{cases} \qquad e\to e' \Rightarrow \begin{cases} \langle v \mid e\rangle \to \langle v \mid e'\rangle \\ v\cdot e \to v\cdot e' \end{cases} \qquad c\to c' \Rightarrow \begin{cases} \mu\beta.c \to \mu\beta.c' \\ \tilde{\mu}x.c \to \tilde{\mu}x.c' \end{cases}$$

We use $\to_{\mathrm{H}}$ for this notion of reduction and $=_{\mathrm{H}}$ for the induced equality.

Notice that rules (λ), (μ), and $(\tilde{\mu})$ reduce commands to commands, rules (η) and $(\eta\mu)$ reduce a term to a term, and rule $(\eta\tilde{\mu})$ reduces a context to a context. Apart from Theorem 5.7, the extensional rules play no role in this paper. Not all commands can be reduced: *e.g.* $\langle x \mid \alpha\rangle$, $\langle \lambda x.v \mid \alpha\rangle$ and $\langle x \mid v\cdot e\rangle$ are irreducible; this is one of the differences between the calculus $\mathcal{X}$, which embodies Gentzen's LK, and $\overline{\lambda}\mu\tilde{\mu}$ [1].

Notice that, although $\overline{\lambda}\mu\tilde{\mu}$ has abstraction, it does not have application, which is natural since LK lacks *elimination rules*. In fact, abstraction's counterpart is that of *context construction* $v\cdot e$, where a term with a hole is built, offering the operand v and the continuation e. The main operators are μ and $\tilde{\mu}$ abstraction, which, in a sense, respectively, correspond to (delayed) substitution (parameter call) and to context call.

Notice that $\overline{\lambda}\mu\tilde{\mu}$ has both *explicit* and *implicit* variables: the implicit variables are for example in $v\cdot e$, where the hole ($\cdot$, which acts as input) does not have an identity, and in $\lambda x.v$ where the context (output) is anonymous. We can make these variables explicit by *naming*, respectively, $\tilde{\mu}y.\langle y \mid v\cdot e\rangle$ and $\mu\alpha.\langle \lambda x.v \mid \alpha\rangle$; in case the variable y (α) does not occur in $v\cdot e$ ($\lambda x.v$), these terms are η redexes, but, in general, the implicit variable can be made to correspond to one that already occurs.

(Implicative) Typing for $\overline{\lambda}\mu\tilde{\mu}$ is defined by:

Definition 5.3 (Typing for $\overline{\lambda}\mu\tilde{\mu}$ [3]) Using Curry types (Definition 1.2), type assignment is defined via the rules:

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

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

$$(\to R):\ \frac{\Gamma,x{:}A\vdash v:B \mid \Delta}{\Gamma\vdash\lambda x.v:A\to B \mid \Delta} \qquad (\to L):\ \frac{\Gamma\vdash v:A \mid \Delta \qquad \Gamma \mid e:B\vdash\Delta}{\Gamma \mid v\cdot e:A\to B\vdash\Delta}$$

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

We write $c:\Gamma\vdash_{\mathrm{H}}\Delta$, $\Gamma\vdash_{\mathrm{H}} v:A \mid \Delta$, and $\Gamma \mid e:A\vdash_{\mathrm{H}}\Delta$ if there exists a derivation built using

these rules that has this judgement in the bottom line.

Observe that $\overline{\lambda}\mu\tilde{\mu}$ has a critical pair in the command $\langle \mu\alpha.c_1 \mid \tilde{\mu}x.c_2 \rangle$, which reduces to both $c_1[\tilde{\mu}x.c_2/\alpha]$ and $c_2[\mu\alpha.c_1/x]$; since *cut*-elimination of the classical sequent calculus is not confluent, neither is reduction in $\overline{\lambda}\mu\tilde{\mu}$. For example, in LK the proof (where (W) is the admissible weakening rule)

$$\dfrac{\dfrac{\boxed{\mathcal{D}_1}}{\dfrac{\Gamma \vdash \Delta}{\Gamma \vdash A,\Delta}\,(W)} \qquad \dfrac{\boxed{\mathcal{D}_2}}{\dfrac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\,(W)}}{\Gamma \vdash \Delta}\,(cut)$$

reduces to both $\mathcal{D}_1$ and $\mathcal{D}_2$, different proofs, albeit for the same sequence; likewise, in $\vdash_{\mathrm{H}}$ we can derive (where α does not appear in c_1, and x does not appear in c_2):

$$\dfrac{\dfrac{\dfrac{\boxed{}}{\dfrac{c_1 : \Gamma \vdash \Delta}{c_1 : \Gamma \vdash \alpha{:}A,\Delta}\,(W)}}{\Gamma \vdash \mu\alpha.c_1 : A \mid \Delta}\,(\mu) \qquad \dfrac{\dfrac{\boxed{}}{\dfrac{c_2 : \Gamma \vdash \Delta}{c_2 : \Gamma, x{:}A \vdash \Delta}\,(W)}}{\Gamma \mid \tilde{\mu}x.c_2 : A \vdash \Delta}\,(\tilde{\mu})}{\langle \mu\alpha.c_1 \mid \tilde{\mu}x.c_2 \rangle : \Gamma \vdash \Delta}\,(cut)$$

and $\langle \mu\alpha.c_1 \mid \tilde{\mu}x.c_2 \rangle$ reduces to both c_1 and c_2, witnesses to the same sequent but not necessarily the same proof.

The $\overline{\lambda}\mu\tilde{\mu}$-calculus expresses the duality of LK's left and right introduction in a very symmetric syntax. But the duality goes beyond that: for instance, the symmetry of the reduction rules displays syntactically the duality between the CBV and CBN evaluations (see also [15]). In [3] the CBV sub-reduction is not defined as a strategy but is obtained by forbidding a $\tilde{\mu}$-reduction when the command is also a μ-redex, whereas the CBN sub-reduction forbids a μ-reduction when the redex is also a $\tilde{\mu}$-redex; there is no other restriction defined in [3, 8] in terms of not permitting certain contextual rules in the definition of CBV and CBN. Since we want CBN and CBV to be reduction *strategies* in the sense that each term has at most one contractable cut, we will give a more detailed definition here.

Definition 5.4 (CBV AND CBN REDUCTION FOR $\overline{\lambda}\mu\tilde{\mu}$) *i)* *Values* V are defined by $V ::= x \mid \lambda x.v$, and *slots*[9] E are defined by $E ::= \alpha \mid v \cdot e$.

ii) CBN-reduction $\to^{\mathrm{N}}_{\mathrm{H}}$ is defined by limiting rule (μ) and restricting the contextual rules:

$$\begin{array}{rrcll} (\lambda): & \langle \lambda x.v_1 \mid v_2 \cdot e \rangle & \to & \langle v_2 \mid \tilde{\mu}x.\langle v_1 \mid e \rangle \rangle & \\ (\mu_{\mathrm{N}}): & \langle \mu\beta.c \mid E \rangle & \to & c[E/\beta] & v \to v' \;\Rightarrow\; \langle v \mid e \rangle \to \langle v' \mid e \rangle \\ (\tilde{\mu}): & \langle v \mid \tilde{\mu}x.c \rangle & \to & c[v/x] & c \to c' \;\Rightarrow\; \mu\beta.c \to \mu\beta.c' \\ (\eta\mu): & \mu\alpha.\langle v \mid \alpha \rangle & \to & v \quad (\alpha \notin fv(v)) & \end{array}$$

iii) CBV-reduction $\to^{\mathrm{v}}_{\mathrm{H}}$ is defined by limiting rule $(\tilde{\mu})$ and restricting the contextual rules:

$$\begin{array}{rrcll} (\lambda): & \langle \lambda x.v_1 \mid v_2 \cdot e \rangle & \to & \langle v_2 \mid \tilde{\mu}x.\langle v_1 \mid e \rangle \rangle & v \to v' \;\Rightarrow\; \begin{cases} \langle v \mid e \rangle \to \langle v' \mid e \rangle \\ v \cdot e \to v' \cdot e \end{cases} \\ (\mu): & \langle \mu\beta.c \mid e \rangle & \to & c[e/\beta] & e \to e' \;\Rightarrow\; \langle V \mid e \rangle \to \langle V \mid e' \rangle \\ (\tilde{\mu}_{\mathrm{v}}): & \langle V \mid \tilde{\mu}x.c \rangle & \to & c[V/x] & c \to c' \;\Rightarrow\; \mu\beta.c \to \mu\beta.c' \end{array}$$

so removes the reduction $\langle \mu\alpha.c \mid \tilde{\mu}x.c' \rangle \to c'[\mu\alpha.c/x]$, and does not permit reduction on the right of the context constructor.

[9] In [8], slots are called *linear evaluation contexts*; we use the terminology 'slot' here because these are contexts with a single hole.

Essentially following [3], an interpretation $[\![\cdot]\!]^{\lambda}$ of the λ-calculus into $\overline{\lambda}\mu\tilde{\mu}$ can be defined as follows:

Definition 5.5 Interpretation of the λ-calculus into $\overline{\lambda}\mu\tilde{\mu}$:

$$\begin{array}{rcl} [\![x]\!]^{\lambda} & \triangleq & x \\ [\![\lambda x.M]\!]^{\lambda} & \triangleq & \lambda x.[\![M]\!]^{\lambda} \\ [\![MN]\!]^{\lambda} & \triangleq & \mu\alpha.\langle [\![M]\!]^{\lambda} \mid [\![N]\!]^{\lambda} \cdot \alpha \rangle \end{array}$$

Notice that λ-values are interpreted by $\overline{\lambda}\mu\tilde{\mu}$-values and that this interpretation is not geared towards a certain reduction strategy.

The interpretation respects term substitution.

Proposition 5.6 ([1]) $[\![M[N/x]]\!]^{\lambda} = [\![M]\!]^{\lambda}[[\![N]\!]^{\lambda}/x]$.

Using this result, we can now show that the interpretation respects β-reduction.

Theorem 5.7 *i) If $M \to_{\beta}^{*} N$, then $[\![M]\!]^{\lambda} \to_{\mathrm{H}}^{*} [\![N]\!]^{\lambda}$.*
ii) If $M \to_{\mathrm{N}}^{*} N$, then $[\![M]\!]^{\lambda} \to_{\mathrm{H}}^{\mathrm{N}*} [\![N]\!]^{\lambda}$.
iii) If $M \to_{\mathrm{V}}^{*} N$, then $[\![M]\!]^{\lambda} \to_{\mathrm{H}}^{\mathrm{V}*} [\![N]\!]^{\lambda}$.

Proof: i) $((\lambda x.M)N \to_{\beta} M[N/x])$:

$$\begin{array}{lll} [\![(\lambda x.M)N]\!]^{\lambda} \triangleq \mu\alpha.\langle [\![\lambda x.M]\!]^{\lambda} \mid [\![N]\!]^{\lambda} \cdot \alpha \rangle & \triangleq & \\ \mu\alpha.\langle \lambda x.[\![M]\!]^{\lambda} \mid [\![N]\!]^{\lambda} \cdot \alpha \rangle \twoheadrightarrow_{\mathrm{H}} (\lambda) & \mu\alpha.\langle [\![N]\!]^{\lambda} \mid \tilde{\mu}x.\langle [\![M]\!]^{\lambda} \mid \alpha \rangle \rangle & \to_{\mathrm{H}} (\tilde{\mu}) \\ \mu\alpha.\langle [\![M]\!]^{\lambda}[[\![N]\!]^{\lambda}/x] \mid \alpha \rangle \twoheadrightarrow_{\mathrm{H}} (\eta\mu) & [\![M]\!]^{\lambda}[[\![N]\!]^{\lambda}/x]. & \end{array}$$

$(M \to_{\beta} N \Rightarrow \lambda x.M \to_{\beta} \lambda x.N)$: $[\![\lambda x.M]\!]^{\lambda} \triangleq \lambda x.[\![M]\!]^{\lambda} \to_{\mathrm{H}}^{*} (IH)\ \lambda x.[\![N]\!]^{\lambda} \triangleq [\![\lambda x.N]\!]^{\lambda}$.
$(M \to_{\beta} N \Rightarrow MP \to_{\beta} NP)$: $[\![MP]\!]^{\lambda} \triangleq \mu\alpha.\langle [\![M]\!]^{\lambda} \mid [\![P]\!]^{\lambda} \cdot \alpha \rangle \to_{\mathrm{H}}^{*} (IH)\ \mu\alpha.\langle [\![N]\!]^{\lambda} \mid [\![P]\!]^{\lambda} \cdot \alpha \rangle \triangleq [\![NP]\!]^{\lambda}$.
$(M \to_{\beta} N \Rightarrow PM \to_{\beta} PN)$: $[\![PM]\!]^{\lambda} \triangleq \mu\alpha.\langle [\![P]\!]^{\lambda} \mid [\![M]\!]^{\lambda} \cdot \alpha \rangle \to_{\mathrm{H}}^{*} (IH)\ \mu\alpha.\langle [\![P]\!]^{\lambda} \mid [\![N]\!]^{\lambda} \cdot \alpha \rangle \triangleq [\![PN]\!]^{\lambda}$.

ii) $((\lambda x.M)N \to_{\mathrm{N}} M[N/x])$: As in part 1 above; notice that all reduction steps are permitted in $\to_{\mathrm{N}}$.
$(M \to_{\mathrm{N}} N \Rightarrow MP \to_{\mathrm{N}} NP)$: $[\![MP]\!]^{\lambda} \triangleq \mu\alpha.\langle [\![M]\!]^{\lambda} \mid [\![P]\!]^{\lambda} \cdot \alpha \rangle \to_{\mathrm{H}}^{\mathrm{N}*} (IH)\ \mu\alpha.\langle [\![N]\!]^{\lambda} \mid [\![P]\!]^{\lambda} \cdot \alpha \rangle \triangleq [\![NP]\!]^{\lambda}$.

iii) $((\lambda x.M)V \to_{\mathrm{V}} M[V/x])$: As in part 1 above; notice that then $N \equiv V$, and the $\tilde{\mu}$-reduction step is permitted.
$(M \to_{\mathrm{V}} N \Rightarrow MP \to_{\mathrm{V}} NP)$: $[\![MP]\!]^{\lambda} \triangleq \mu\alpha.\langle [\![M]\!]^{\lambda} \mid [\![P]\!]^{\lambda} \cdot \alpha \rangle \to_{\mathrm{H}}^{\mathrm{V}*} (IH)\ \mu\alpha.\langle [\![N]\!]^{\lambda} \mid [\![P]\!]^{\lambda} \cdot \alpha \rangle \triangleq [\![NP]\!]^{\lambda}$.
$(M \to_{\mathrm{V}} N \Rightarrow VM \to_{\mathrm{V}} VN)$: $[\![VM]\!]^{\lambda} \triangleq \mu\alpha.\langle [\![V]\!]^{\lambda} \mid [\![M]\!]^{\lambda} \cdot \alpha \rangle \to_{\mathrm{H}}^{\mathrm{V}*} (IH)\ \mu\alpha.\langle [\![V]\!]^{\lambda} \mid [\![N]\!]^{\lambda} \cdot \alpha \rangle \triangleq [\![VN]\!]^{\lambda}$. □

This interpretation corresponds to running the λ-calculus in a stack-based CBN abstract machine (like Krivine's machine [9]): evaluating the term $\lambda x.M$ takes a term off the stack and substitutes it for x in M. For the application MN, N gets placed onto the stack and the machine then attempts to run M to a term of the form $\lambda x.M'$. So we can think of it as a function that takes some existing stack, and returns the machine which runs M in the existing stack with N pushed on top.

Example 5.8 In $\overline{\lambda}\mu\tilde{\mu}$ we express the interaction between a program (term) and its context via commands. Although there is no notion of application, $\overline{\lambda}\mu\tilde{\mu}$ sees $[\![MN_1 \cdots N_n]\!]^{\lambda}$ as running $[\![M]\!]^{\lambda}$ in the context that offers the terms $[\![N_1]\!]^{\lambda}, \ldots, [\![N_n]\!]^{\lambda}$ in sequence. To understand this, first notice that

$$\begin{array}{rcll} [\![MN_1N_2]\!]^{\lambda} & \triangleq & \mu\alpha.\langle [\![MN_1]\!]^{\lambda} \mid [\![N_2]\!]^{\lambda} \cdot \alpha\rangle \\ & \triangleq & \mu\alpha.\langle \mu\beta.\langle [\![M]\!]^{\lambda} \mid [\![N_1]\!]^{\lambda} \cdot \beta\rangle \mid [\![N_2]\!]^{\lambda} \cdot \alpha\rangle \\ & \to_{\mathrm{H}} (\mu) & \mu\alpha.\langle [\![M]\!]^{\lambda} \mid [\![N_1]\!]^{\lambda} \cdot [\![N_2]\!]^{\lambda} \cdot \alpha\rangle \end{array}$$

so it is easy to verify that

$$[\![MN_1 \cdots N_n]\!]^{\lambda} \ \to^{*} (\mu) \ \ \mu\alpha.\langle [\![M]\!]^{\lambda} \mid [\![N_1]\!]^{\lambda} \cdot \ldots \cdot [\![N_n]\!]^{\lambda} \cdot \alpha\rangle$$

which puts into evidence that, for λ-terms, the only contexts that are needed are *stacks*. Notice that the context $[\![N_1]\!]^{\lambda} \cdot ([\![N_1]\!]^{\lambda} \cdot \ldots \cdot [\![N_n]\!]^{\lambda} \cdot \alpha)$ represents the λ-context for M, so stands for $\mathsf{C}[\lceil\rfloor N_1]$, where $\mathsf{C}[\,] \equiv \lceil\rfloor N_2 \cdots N_n$.

6 Mapping eCBPV in $\overline{\lambda}\mu\tilde{\mu}$

In this section we will show that we can interpret eCBPV into the $\overline{\lambda}\mu\tilde{\mu}$-calculus and simulate it via the $\overline{\lambda}\mu\tilde{\mu}$ reduction rules.

Definition 6.1 (Direct interpretation) The interpretation $[\![\cdot]\!]^{\mathrm{e}}$ of eCBPV-values and computations into $\overline{\lambda}\mu\tilde{\mu}$-terms is defined as:

$$\begin{array}{rcl} [\![y]\!]^{\mathrm{e}} & = & y \\ [\![\{M\}]\!]^{\mathrm{e}} & = & [\![M]\!]^{\mathrm{e}} \\ [\![x!]\!]^{\mathrm{e}} & = & x \end{array} \qquad \begin{array}{rcl} [\![\lambda x.M]\!]^{\mathrm{e}} & = & \lambda x.[\![M]\!]^{\mathrm{e}} \\ [\![MV]\!]^{\mathrm{e}} & = & \mu\alpha.\langle [\![M]\!]^{\mathrm{e}} \mid [\![V]\!]^{\mathrm{e}} \cdot \alpha\rangle \\ [\![ret\, V]\!]^{\mathrm{e}} & = & [\![V]\!]^{\mathrm{e}} \\ [\![x := M; N]\!]^{\mathrm{e}} & = & \mu\alpha.\langle [\![M]\!]^{\mathrm{e}} \mid \tilde{\mu}x.\langle [\![N]\!]^{\mathrm{e}} \mid \alpha\rangle\rangle \end{array}$$

At the moment it is unclear if this encoding can be extended into one for CBPV: to model the reduction rule (U), it seems necessary to extend $\overline{\lambda}\mu\tilde{\mu}$ syntactically as well with features that represent thunking and forcing.

We can show that the interpretation respects term substitution.

Lemma 6.2 $[\![M\langle V/x\rangle]\!]^{e} = [\![M]\!]^{e}[[\![V]\!]^{e}/x]$.

Proof: By induction on the definition of $\cdot\langle V/x\rangle$. We only show the base cases, the others follow by straightforward induction.

$(x\langle V/x\rangle = V)$: We have two cases:
- $(V = y)$: $[\![x\langle y/x\rangle]\!]^{\mathrm{e}} = [\![y]\!]^{\mathrm{e}} = y = x[y/x] = [\![x]\!]^{\mathrm{e}}[[\![y]\!]^{\mathrm{e}}/x]$.
- $(V = \{N\})$: $[\![x\langle\{N\}/x\rangle]\!]^{\mathrm{e}} = [\![\{N\}]\!]^{\mathrm{e}} = [\![N]\!]^{\mathrm{e}} = x[[\![N]\!]^{\mathrm{e}}/x] = [\![x]\!]^{\mathrm{e}}[[\![\{N\}]\!]^{\mathrm{e}}/x]$.

$(z\langle V/x\rangle = z)$: $[\![z\langle V/x\rangle]\!]^{\mathrm{e}} = [\![z]\!]^{\mathrm{e}} = z = z[[\![V]\!]^{\mathrm{e}}/x] = [\![z]\!]^{\mathrm{e}}[[\![V]\!]^{\mathrm{e}}/x]$.

$(x!\langle y/x\rangle = y!)$: $[\![x!\langle y/x\rangle]\!]^{\mathrm{e}} = [\![y!]\!]^{\mathrm{e}} = y = x[y/x] = [\![x!]\!]^{\mathrm{e}}[[\![y]\!]^{\mathrm{e}}/x]$.

$(x!\langle\{N\}/x\rangle = N)$: $[\![x!\langle\{N\}/x\rangle]\!]^{\mathrm{e}} = [\![N]\!]^{\mathrm{e}} = x[[\![N]\!]^{\mathrm{e}}/x] = [\![x!]\!]^{\mathrm{e}}[[\![\{N\}]\!]^{\mathrm{e}}/x]$.

$(z!\langle y/x\rangle = z!)$: $[\![z!\langle y/x\rangle]\!]^{\mathrm{e}} = [\![z!]\!]^{\mathrm{e}} = z = z[[\![y]\!]^{\mathrm{e}}/x] = [\![z!]\!]^{\mathrm{e}}[[\![y]\!]^{\mathrm{e}}/x]$.

$(z!\langle\{N\}/x\rangle = z!)$: $[\![z!\langle\{N\}/x\rangle]\!]^{\mathrm{e}} = [\![z!]\!]^{\mathrm{e}} = z = z[[\![\{N\}]\!]^{\mathrm{e}}/x] = [\![z!]\!]^{\mathrm{e}}[[\![\{N\}]\!]^{\mathrm{e}}/x]$. □

We can now show that reduction in eCBPV is preserved under the interpretation.

Theorem 6.3 *If $M \to_e N$, then $[\![M]\!]^{e} \to_{\mathrm{H}}^{\mathrm{N}*} [\![N]\!]^{e}$.*

Proof: By induction on the definition of $\to_{\mathrm{e}}$.

$$\begin{array}{lll} ((\lambda x.M)V \to M\langle V/x\rangle): & [\![(\lambda x.M)\,V]\!]^{\mathrm{e}} = \mu\alpha.\langle\lambda x.[\![M]\!]^{\mathrm{e}} \mid [\![V]\!]^{\mathrm{e}} \cdot \alpha\rangle & \to_{\mathrm{H}}^{\mathrm{N}} (\lambda) \\ & \mu\alpha.\langle [\![V]\!]^{\mathrm{e}} \mid \tilde{\mu}x.\langle [\![M]\!]^{\mathrm{e}} \mid \alpha\rangle\rangle \ \to_{\mathrm{H}}^{\mathrm{N}} (\tilde{\mu}) \ \ \mu\alpha.\langle [\![M]\!]^{\mathrm{e}}[[\![V]\!]^{\mathrm{e}}/x] \mid \alpha\rangle & \to_{\mathrm{H}}^{\mathrm{N}} (\eta\mu) \\ & [\![M]\!]^{\mathrm{e}}[[\![V]\!]^{\mathrm{e}}/x] = (6.2) \ \ [\![M\langle V/x\rangle]\!]^{\mathrm{e}} & \end{array}$$

$(x := \mathit{ret}\, V; N \to N\langle V/x\rangle)$: $[\![x := \mathit{ret}\, V; N]\!]^{\mathrm{e}}_{\shortparallel} = \mu\alpha.\langle [\![V]\!]^{\mathrm{e}}_{\shortparallel} \mid \tilde{\mu}x.\langle [\![N]\!]^{\mathrm{e}}_{\shortparallel} \mid \alpha\rangle\rangle \to^{\mathrm{N}}_{\mathrm{H}} (\tilde{\mu})$
$\mu\alpha.\langle [\![N]\!]^{\mathrm{e}}_{\shortparallel}[[\![V]\!]^{\mathrm{e}}_{\shortparallel}/x] \mid \alpha\rangle \to^{\mathrm{N}}_{\mathrm{H}} (\eta\mu)\ [\![N]\!]^{\mathrm{e}}_{\shortparallel}[[\![V]\!]^{\mathrm{e}}_{\shortparallel}/x] =_{(6.2)} [\![N\langle V/x\rangle]\!]^{\mathrm{e}}_{\shortparallel}$

$(M \to_{\mathrm{P}} N \Rightarrow MV \to_{\mathrm{P}} NV)$: $[\![MV]\!]^{\mathrm{e}}_{\shortparallel} = \mu\alpha.\langle [\![M]\!]^{\mathrm{e}}_{\shortparallel} \mid [\![V]\!]^{\mathrm{e}}_{\shortparallel} \cdot \alpha\rangle \to^{\mathrm{N}*}_{\mathrm{H}} (IH)\ \mu\alpha.\langle [\![N]\!]^{\mathrm{e}}_{\shortparallel} \mid [\![V]\!]^{\mathrm{e}}_{\shortparallel} \cdot \alpha\rangle = [\![NV]\!]^{\mathrm{e}}_{\shortparallel}$

$(M \to_{\mathrm{P}} N \Rightarrow x := M; P \to x := N; P)$: $[\![x := M; P]\!]^{\mathrm{e}}_{\shortparallel} = \mu\alpha.\langle [\![M]\!]^{\mathrm{e}}_{\shortparallel} \mid \tilde{\mu}x.\langle [\![P]\!]^{\mathrm{e}}_{\shortparallel} \mid \alpha\rangle\rangle \to^{\mathrm{N}*}_{\mathrm{H}} (IH)$
$\mu\alpha.\langle [\![N]\!]^{\mathrm{e}}_{\shortparallel} \mid \tilde{\mu}x.\langle [\![P]\!]^{\mathrm{e}}_{\shortparallel} \mid \alpha\rangle\rangle = [\![x := N; P]\!]^{\mathrm{e}}_{\shortparallel}$ □

This stresses again that reduction in CBPV is essentially CBN; moreover, all redexes are unique and reductions deterministic, which leads to:

Theorem 6.4 (Full abstraction) *If $[\![M]\!]^{e}_{\shortparallel} \to_{e} Q$, then there exists $N \in$ CBPV such that $M \to_{e} N$, and $Q \to^{\mathrm{N}*}_{\mathrm{H}} [\![N]\!]^{e}_{\shortparallel}$.*

We can show a similar result for the abstract machine CK.

Definition 6.5 The interpretation of evaluation stacks and configurations in CK is defined through:

$$
\begin{array}{rcl@{\qquad}rcl}
[\![\epsilon]\!]^{\mathrm{CK}}_{\shortparallel\alpha} & = & \alpha & [\![\langle MV \mid S\rangle]\!]^{\mathrm{CK}}_{\shortparallel} & = & \mu\alpha.\langle [\![M]\!]^{\mathrm{CK}}_{\shortparallel} \mid [\![V]\!]^{\mathrm{CK}}_{\shortparallel} \cdot [\![S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle \\
[\![V : S]\!]^{\mathrm{CK}}_{\shortparallel\alpha} & = & [\![V]\!]^{\mathrm{CK}}_{\shortparallel} \cdot [\![S]\!]^{\mathrm{CK}}_{\shortparallel\alpha} & [\![\langle x := M; N \mid S\rangle]\!]^{\mathrm{CK}}_{\shortparallel} & = & \mu\alpha.\langle [\![M]\!]^{\mathrm{CK}}_{\shortparallel} \mid \tilde{\mu}x.\langle [\![N]\!]^{\mathrm{CK}}_{\shortparallel} \mid [\![S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle\rangle \\
[\![x := [\,]; N : S]\!]^{\mathrm{CK}}_{\shortparallel\alpha} & = & \tilde{\mu}x.\langle [\![N]\!]^{\mathrm{CK}}_{\shortparallel} \mid [\![S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle & [\![\langle M \mid S\rangle]\!]^{\mathrm{CK}}_{\shortparallel} & = & \mu\alpha.\langle [\![M]\!]^{\mathrm{CK}}_{\shortparallel} \mid [\![S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle \quad (\textit{otherwise})
\end{array}
$$

In fact, it was this similarity between stacks in CK and (applicative) contexts of $\overline{\lambda}\mu\tilde{\mu}$ that was the inspiration for this paper.

We can now show that this interpretation respects $\to_{\mathrm{CK}}$-reductions.

Lemma 6.6 If $\langle M \mid S\rangle \to_{\mathrm{CK}} \langle M \mid S\rangle$, then $[\![\langle M \mid S\rangle]\!]^{\mathrm{CK}}_{\shortparallel} \to^{\mathrm{N}*}_{\mathrm{H}} [\![\langle M \mid S\rangle]\!]^{\mathrm{CK}}_{\shortparallel}$.

Proof: $(\langle \lambda x.M \mid V : S\rangle \to_{\mathrm{CK}} \langle M[V/x] \mid S\rangle)$: $[\![\langle \lambda x.M \mid V : S\rangle]\!]^{\mathrm{CK}}_{\shortparallel} =$
$\mu\alpha.\langle \lambda x.[\![M]\!]^{\mathrm{CK}}_{\shortparallel} \mid [\![V]\!]^{\mathrm{CK}}_{\shortparallel} \cdot [\![S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle \to^{\mathrm{N}}_{\mathrm{H}} \mu\alpha.\langle [\![V]\!]^{\mathrm{CK}}_{\shortparallel} \mid \tilde{\mu}x.\langle [\![M]\!]^{\mathrm{CK}}_{\shortparallel} \mid [\![S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle\rangle \to^{\mathrm{N}}_{\mathrm{H}}$
$\mu\alpha.\langle [\![M]\!]^{\mathrm{CK}}_{\shortparallel}[[\![V]\!]^{\mathrm{CK}}_{\shortparallel}/x] \mid [\![S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle = \mu\alpha.\langle [\![M[V/x]]\!]^{\mathrm{CK}}_{\shortparallel} \mid [\![S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle = [\![\langle M[V/x] \mid S\rangle]\!]^{\mathrm{CK}}_{\shortparallel}$

$(\langle MV \mid S\rangle \to_{\mathrm{CK}} \langle M \mid V : S\rangle)$: $[\![\langle MV \mid S\rangle]\!]^{\mathrm{CK}}_{\shortparallel} = \mu\alpha.\langle [\![M]\!]^{\mathrm{CK}}_{\shortparallel} \mid [\![V]\!]^{\mathrm{CK}}_{\shortparallel} \cdot [\![S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle = [\![\langle M \mid V : S\rangle]\!]^{\mathrm{CK}}_{\shortparallel}$

$(\langle x := M; N \mid S\rangle \to_{\mathrm{CK}} \langle M \mid x := [\,]; N : S\rangle)$: $[\![\langle x := M; N \mid S\rangle]\!]^{\mathrm{CK}}_{\shortparallel} =$
$\mu\alpha.\langle [\![M]\!]^{\mathrm{CK}}_{\shortparallel} \mid \tilde{\mu}x.\langle [\![N]\!]^{\mathrm{CK}}_{\shortparallel} \mid [\![S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle\rangle = \mu\alpha.\langle [\![M]\!]^{\mathrm{CK}}_{\shortparallel} \mid [\![x := [\,]; N : S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle = [\![\langle M \mid x := [\,]; N : S\rangle]\!]^{\mathrm{CK}}_{\shortparallel}$

$(\langle \mathit{ret}\, V \mid x := [\,]; M : S\rangle \to_{\mathrm{CK}} \langle M[V/x] \mid S\rangle)$: $[\![\langle \mathit{ret}\, V \mid x := [\,]; M : S\rangle]\!]^{\mathrm{CK}}_{\shortparallel} =$
$\mu\alpha.\langle [\![V]\!]^{\mathrm{CK}}_{\shortparallel} \mid [\![x := [\,]; M : S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle = \mu\alpha.\langle [\![V]\!]^{\mathrm{CK}}_{\shortparallel} \mid \tilde{\mu}x.\langle [\![M]\!]^{\mathrm{CK}}_{\shortparallel} \mid [\![S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle\rangle \to^{\mathrm{N}}_{\mathrm{H}}$
$\mu\alpha.\langle [\![M]\!]^{\mathrm{CK}}_{\shortparallel}[[\![V]\!]^{\mathrm{CK}}_{\shortparallel}/x] \mid [\![S]\!]^{\mathrm{CK}}_{\shortparallel\alpha}\rangle = [\![\langle M[V/x] \mid S\rangle]\!]^{\mathrm{CK}}_{\shortparallel}$ □

Using the 'inverse' of the type interpretation from Definition 4.11, we can also show that type assignment is preserved under the interpretations.

Definition 6.7 The type interpretation $[\![\cdot]\!]^{\mathrm{e}}_{\shortparallel}$ is defined as:

$$
\begin{array}{rcl}
[\![\varphi]\!]^{\mathrm{e}}_{\shortparallel} & = & \varphi \\
[\![\mathsf{U}\mathsf{A}]\!]^{\mathrm{e}}_{\shortparallel} & = & [\![\mathsf{A}]\!]^{\mathrm{e}}_{\shortparallel} \\
[\![A \to \mathsf{B}]\!]^{\mathrm{e}}_{\shortparallel} & = & [\![A]\!]^{\mathrm{e}}_{\shortparallel} \to [\![\mathsf{B}]\!]^{\mathrm{e}}_{\shortparallel} \\
[\![\mathsf{F}A]\!]^{\mathrm{e}}_{\shortparallel} & = & [\![A]\!]^{\mathrm{e}}_{\shortparallel}
\end{array}
$$

and the environment interpretation as:

$$[\![\Gamma]\!]^{\mathrm{e}}_{\shortparallel} = \{\, x{:}[\![A]\!]^{\mathrm{e}}_{\shortparallel} \mid x{:}A \in \Gamma \,\}$$

We can now show that $\overline{\lambda}\mu\tilde{\mu}$ is suited to not just model the kind of reduction of CBPV, but also its type assignment.

Theorem 6.8 *i) If* $\Gamma \vdash_{\mathrm{V}} V : A$, *then* $[\![\Gamma]\!]^{\mathrm{e}} \vdash_{\mathrm{H}} [\![V]\!]^{\mathrm{e}} : [\![A]\!]^{\mathrm{e}} \mid$.

ii) If $\Gamma \vdash_{\mathrm{C}} M : \mathsf{A}$, *then* $[\![\Gamma]\!]^{\mathrm{e}} \vdash_{\mathrm{H}} [\![M]\!]^{\mathrm{e}} : [\![\mathsf{A}]\!]^{\mathrm{e}} \mid$.

Proof: By induction on definition of type assignment for eCBPV-terms.

(*axiom*): Then $V \equiv x$, and $\Gamma = \Gamma', x{:}A$; since $[\![x]\!]^{\mathrm{e}} \triangleq x$, and $[\![\Gamma]\!]^{\mathrm{e}} = [\![\Gamma']\!]^{\mathrm{e}}, x{:}[\![A]\!]^{\mathrm{e}}$, by rule $(Ax\text{-}R)$ also $[\![\Gamma]\!]^{\mathrm{e}} \vdash_{\mathrm{H}} [\![v]\!]^{\mathrm{e}} : [\![A]\!]^{\mathrm{e}} \mid$.

(*thunk*): Then $A = \mathsf{U}\mathsf{A}$, $M \equiv \{N\}$, and $\Gamma \vdash_{\mathrm{C}} N : \mathsf{A}$; by induction we have $[\![\Gamma]\!]^{\mathrm{e}} \vdash_{\mathrm{H}} [\![N]\!]^{\mathrm{e}} : [\![\mathsf{A}]\!]^{\mathrm{e}} \mid$. Notice that $[\![\mathsf{U}\mathsf{A}]\!]^{\mathrm{e}} = [\![\mathsf{A}]\!]^{\mathrm{e}}$ and $[\![\{N\}]\!]^{\mathrm{e}} = [\![N]\!]^{\mathrm{e}}$.

(*abstr*): Then $\mathsf{A} = B \to \mathsf{C}$, $M \equiv \lambda x.N$, and $\Gamma, x{:}B \vdash_{\mathrm{C}} N : \mathsf{C}$; since $[\![\Gamma, x{:}B]\!]^{\mathrm{e}} = [\![\Gamma]\!]^{\mathrm{e}}, x{:}[\![B]\!]^{\mathrm{e}}$, by induction we have $[\![\Gamma]\!]^{\mathrm{e}}, x{:}[\![B]\!]^{\mathrm{e}} \vdash_{\mathrm{H}} [\![N]\!]^{\mathrm{e}} : [\![\mathsf{C}]\!]^{\mathrm{e}} \mid$. We can construct:

$$\dfrac{\boxed{}}{[\![\Gamma]\!]^{\mathrm{f}}, x{:}[\![B]\!]^{\mathrm{f}} \vdash_{\mathrm{H}} [\![N]\!]^{\mathrm{f}} : [\![\mathsf{C}]\!]^{\mathrm{f}} \mid}$$
$$\dfrac{}{[\![\Gamma]\!]^{\mathrm{f}} \vdash_{\mathrm{H}} \lambda x.[\![N]\!]^{\mathrm{f}} : [\![B]\!]^{\mathrm{f}} \to [\![\mathsf{C}]\!]^{\mathrm{f}} \mid}\ (\to R)$$

Notice that $[\![\lambda x.M]\!]^{\mathrm{e}} \triangleq \lambda x.[\![N]\!]^{\mathrm{e}}$ and $[\![\mathsf{A}]\!]^{\mathrm{e}} = [\![B]\!]^{\mathrm{e}} \to [\![\mathsf{C}]\!]^{\mathrm{e}}$.

(*appl*): Then $M \equiv NV$, and there exists B such that $\Gamma \vdash_{\mathrm{C}} u : B \to \mathsf{A}$ and $\Gamma \vdash_{\mathrm{V}} v : B$. Then, by induction, $[\![\Gamma]\!]^{\mathrm{e}} \vdash_{\mathrm{H}} [\![N]\!]^{\mathrm{e}} : [\![B \to \mathsf{A}]\!]^{\mathrm{e}} \mid$ and $[\![\Gamma]\!]^{\mathrm{e}} \vdash_{\mathrm{H}} [\![V]\!]^{\mathrm{e}} : [\![B]\!]^{\mathrm{e}} \mid$. We can construct (with $\Gamma' = [\![\Gamma]\!]^{\mathrm{e}}, x{:}\mathsf{U}(B \to \mathsf{A}), y{:}B$):

$$\dfrac{\dfrac{\dfrac{\boxed{}}{[\![\Gamma]\!]^{\mathrm{f}} \vdash [\![N]\!]^{\mathrm{f}} : [\![B \to \mathsf{A}]\!]^{\mathrm{f}} \mid}}{[\![\Gamma]\!]^{\mathrm{f}} \vdash [\![N]\!]^{\mathrm{f}} : [\![B \to \mathsf{A}]\!]^{\mathrm{f}} \mid \alpha : [\![\mathsf{A}]\!]^{\mathrm{f}}}\ (Wk) \qquad \dfrac{\dfrac{\boxed{}}{[\![\Gamma]\!]^{\mathrm{f}} \vdash [\![V]\!]^{\mathrm{f}} : [\![B]\!]^{\mathrm{f}} \mid}}{[\![\Gamma]\!]^{\mathrm{f}} \vdash [\![V]\!]^{\mathrm{f}} : [\![B]\!]^{\mathrm{f}} \mid \alpha : [\![\mathsf{A}]\!]^{\mathrm{f}}}\ (Wk) \qquad \dfrac{}{[\![\Gamma]\!]^{\mathrm{f}} \mid \alpha : [\![\mathsf{A}]\!]^{\mathrm{f}} \vdash \alpha : [\![\mathsf{A}]\!]^{\mathrm{f}}}\ (Ax\text{-}L)}{[\![\Gamma]\!]^{\mathrm{f}} \mid [\![V]\!]^{\mathrm{f}} \cdot \alpha : [\![B \to \mathsf{A}]\!]^{\mathrm{f}} \vdash \alpha : [\![\mathsf{A}]\!]^{\mathrm{f}}}\ (\to L)}\ (cut)}{\dfrac{\langle [\![M]\!]^{\mathrm{f}} \mid [\![V]\!]^{\mathrm{f}} \cdot \alpha \rangle : [\![\Gamma]\!]^{\mathrm{f}} \vdash \alpha : [\![\mathsf{A}]\!]^{\mathrm{f}}}{[\![\Gamma]\!]^{\mathrm{f}} \vdash \mu\alpha.\langle [\![N]\!]^{\mathrm{f}} \mid [\![V]\!]^{\mathrm{f}} \cdot \alpha \rangle : [\![\mathsf{A}]\!]^{\mathrm{f}} \mid}\ (\mu)}$$

Notice that $[\![B \to \mathsf{A}]\!]^{\mathrm{e}} = [\![B]\!]^{\mathrm{e}} \to [\![\mathsf{A}]\!]^{\mathrm{e}}$ so the step $(\to L)$ is justified, and $[\![NV]\!]^{\mathrm{e}} \triangleq \mu\alpha.\langle [\![N]\!]^{\mathrm{e}} \mid [\![V]\!]^{\mathrm{e}} \cdot \alpha \rangle$.

(*force*): Then $M = x!$, and $\Gamma \vdash_{\mathrm{V}} x : \mathsf{U}\,\mathsf{A}$; notice that $[\![x!]\!]^{\mathrm{e}} = x$ and $[\![\mathsf{U}\mathsf{A}]\!]^{\mathrm{e}} = [\![A]\!]^{\mathrm{e}}$. Then $x{:}\mathsf{U}\mathsf{A} \in \Gamma$, so $x{:}[\![A]\!]^{\mathrm{e}} \in [\![\Gamma]\!]^{\mathrm{e}}$, and by rule $(Ax\text{-}R)$ also $[\![\Gamma]\!]^{\mathrm{e}} \vdash_{\mathrm{H}} [\![x]\!]^{\mathrm{e}} : [\![A]\!]^{\mathrm{e}} \mid$.

(*ret*): Then $\mathsf{A} = \mathsf{F}A$, $M = ret\,V$, and $\Gamma \vdash_{\mathrm{V}} V : A$; notice that $[\![ret\,V]\!]^{\mathrm{e}} = [\![V]\!]^{\mathrm{e}}$ and $[\![\mathsf{F}A]\!]^{\mathrm{e}} = [\![A]\!]^{\mathrm{e}}$. Then by induction $[\![\Gamma]\!]^{\mathrm{e}} \vdash_{\mathrm{H}} [\![V]\!]^{\mathrm{e}} : [\![A]\!]^{\mathrm{e}} \mid$.

(*assign*): Then $M = x := P; Q$, and there exists x and B such that both $\Gamma \vdash_{\mathrm{C}} P : \mathsf{F}B$ and $\Gamma, x{:}B \vdash_{\mathrm{C}} Q : \mathsf{A}$. Then, by induction, we have $[\![\Gamma]\!]^{\mathrm{e}} \vdash_{\mathrm{H}} [\![P]\!]^{\mathrm{e}} : [\![\mathsf{F}B]\!]^{\mathrm{e}} \mid$ and $[\![\Gamma, x{:}B]\!]^{\mathrm{e}} \vdash_{\mathrm{H}} [\![Q]\!]^{\mathrm{e}} : [\![\mathsf{A}]\!]^{\mathrm{e}} \mid$. Notice that $[\![\Gamma, x{:}B]\!]^{\mathrm{e}} = [\![\Gamma]\!]^{\mathrm{e}}, x{:}[\![B]\!]^{\mathrm{e}}$, $[\![\mathsf{F}B]\!]^{\mathrm{e}} = [\![B]\!]^{\mathrm{e}}$, and $[\![\mathsf{A}]\!]^{\mathrm{e}} = [\![A]\!]^{\mathrm{e}}$.

$$\dfrac{\dfrac{\dfrac{\boxed{}}{[\![\Gamma]\!]^{\mathrm{f}} \vdash [\![P]\!]^{\mathrm{f}} : [\![B]\!]^{\mathrm{f}} \mid}}{[\![\Gamma]\!]^{\mathrm{f}} \vdash [\![P]\!]^{\mathrm{f}} : [\![B]\!]^{\mathrm{f}} \mid \alpha : [\![\mathsf{A}]\!]^{\mathrm{f}}}\ (Wk) \qquad \dfrac{\dfrac{\dfrac{\boxed{}}{[\![\Gamma]\!]^{\mathrm{f}}, x{:}[\![B]\!]^{\mathrm{f}} \vdash [\![Q]\!]^{\mathrm{f}} : [\![\mathsf{A}]\!]^{\mathrm{f}} \mid}}{[\![\Gamma]\!]^{\mathrm{f}}, x{:}[\![B]\!]^{\mathrm{f}} \vdash [\![Q]\!]^{\mathrm{f}} : [\![\mathsf{A}]\!]^{\mathrm{f}} \mid \alpha : [\![\mathsf{A}]\!]^{\mathrm{f}}}\ (Wk) \qquad \dfrac{}{[\![\Gamma]\!]^{\mathrm{f}}, x{:}[\![B]\!]^{\mathrm{f}} \mid \alpha : [\![\mathsf{A}]\!]^{\mathrm{f}} \vdash \alpha : [\![\mathsf{A}]\!]^{\mathrm{f}}}\ (Ax\text{-}L)}{\langle [\![Q]\!]^{\mathrm{f}} \mid \alpha \rangle : [\![\Gamma]\!]^{\mathrm{f}}, x{:}[\![B]\!]^{\mathrm{f}} \vdash \alpha : [\![\mathsf{A}]\!]^{\mathrm{f}}}\ (cut)}{[\![\Gamma]\!]^{\mathrm{f}} \mid \tilde{\mu}x.\langle [\![Q]\!]^{\mathrm{f}} \mid \alpha \rangle : [\![B]\!]^{\mathrm{f}} \vdash \alpha : [\![\mathsf{A}]\!]^{\mathrm{f}}}\ (\tilde{\mu})}\ (cut)}{\dfrac{\langle [\![P]\!]^{\mathrm{f}} \mid \tilde{\mu}x.\langle [\![Q]\!]^{\mathrm{f}} \mid \alpha \rangle \rangle : [\![\Gamma]\!]^{\mathrm{f}} \vdash \alpha : [\![\mathsf{A}]\!]^{\mathrm{f}}}{[\![\Gamma]\!]^{\mathrm{f}} \vdash \mu\alpha.\langle [\![P]\!]^{\mathrm{f}} \mid \tilde{\mu}x.\langle [\![Q]\!]^{\mathrm{f}} \mid \alpha \rangle \rangle : [\![\mathsf{A}]\!]^{\mathrm{f}} \mid}\ (\mu)}$$

and $[\![x := P; Q]\!]^{\mathrm{e}} = \mu\alpha.\langle [\![P]\!]^{\mathrm{e}} \mid \tilde{\mu}x.\langle [\![Q]\!]^{\mathrm{e}} \mid \alpha \rangle \rangle$. □

Conclusion

We have shown that, although partially presented for that purpose, Levy's CBPV calculus is not suitable to represent the CBN or CBV reduction of the λ-calculus and that in order to represent that kind of reduction, it is necessary to change the way term substitution is defined. This is done in *essential* CBPV, which we defined here; since now we never create forcing of thunked terms, also the *unblock* reduction rule is removed.

For this restricted version of CBPV we have shown that we can fully represent the CBN or CBV reduction of the λ-calculus, as well as preserve typeability. Moreover, we defined a mapping into Curien and Herbelin's $\overline{\lambda}\mu\tilde{\mu}$, and showed that reduction in eCBPV can be successfully modeled in the CBN partition of $\overline{\lambda}\mu\tilde{\mu}$, as well as that typeability is preserved.

References

[1] S. van Bakel and P. Lescanne. Computation with Classical Sequents. *Mathematical Structures in Computer Science*, 18:555–609, 2008.

[2] H. Barendregt. *The Lambda Calculus: its Syntax and Semantics*. North-Holland, Amsterdam, revised edition, 1984.

[3] P.-L. Curien and H. Herbelin. The Duality of Computation. In *Proceedings of the 5th ACM SIGPLAN International Conference on Functional Programming (ICFP'00)*, volume 35.9 of *ACM Sigplan Notices*, pages 233–243. ACM, 2000.

[4] H.B. Curry. Grundlagen der Kombinatorischen Logik. *American Journal of Mathematics*, 52:509–536, 789–834, 1930.

[5] Yannick Forster, Steven Schäfer, Simon Spies, and Kathrin Stark. Call-by-Push-Value in Coq: Operational, Equational, and Denotational Theory. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 118–131, New York, NY, USA, 2019. Association for Computing Machinery.

[6] G. Gentzen. Untersuchungen über das Logische Schliessen. *Mathematische Zeitschrift*, 39(2):176–210 and 405–431, 1935.

[7] H. Herbelin. *Séquents qu'on calcule : de l'interprétation du calcul des séquents comme calcul de λ-termes et comme calcul de stratégies gagnantes*. Thèse d'université, Université Paris 7, Janvier 1995.

[8] H. Herbelin. On the Degeneracy of Sigma-Types in Presence of Computational Classical Logic. In P. Urzyczyn, editor, *Typed Lambda Calculi and Applications, 7th International Conference, TLCA 2005, Nara, Japan, April 21-23, 2005, Proceedings*, volume 3461 of *Lecture Notes in Computer Science*, pages 209–220. Springer, 2005.

[9] J-L. Krivine. A call-by-name lambda-calculus machine. *Higher Order and Symbolic Computation*, 20(3):199–207, 2007.

[10] Paul Blain Levy. Call-By-Push-Value: Decomposing Call-By-Value and Call-By-Name. *Higher Order Symbolic Computation*, 19(4):377–414, 2006.

[11] P.B. Levy. Call-by-Push-Value: A Subsuming Paradigm. In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications*, pages 228–243, Berlin, Heidelberg, 1999. Springer.

[12] P.B. Levy. *Call-By-Push-Value: A Functional/Imperative Synthesis (Semantics Structures in Computation, V. 2)*. Kluwer Academic Publishers, USA, 2001.

[13] M. Parigot. Classical Proofs as Programs. In *Kurt Gödel Colloquium*, pages 263–276, 1993. Presented at TYPES Workshop, at Bǎstad, June 1992.

[14] José Espírito Santo. The Polarized λ-calculus. In V. Nigam and M. Florido, editors, *11th Workshop on Logical and Semantic Frameworks with Applications, LSFA 2016, Porto, Portugal, January 1, 2016*, volume 332 of *Electronic Notes in Theoretical Computer Science*, pages 149–168, 2016.

[15] P. Wadler. Call-by-Value is Dual to Call-by-Name. In *Proceedings of the eighth ACM SIGPLAN international conference on Functional programming*, pages 189 – 201, 2003.