# A Calculus of Delayed Reductions

Steffen van Bakel, Emma Tye, and Nicholas Wu

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

**Abstract**

We introduce the Calculus of Delayed Reduction (CDR), that expresses that redexes can only be contracted when brought to the right position in a term, and will show that Call by Name or Value (CBN, CBV) reduction for the $\lambda$-calculus can be modelled through reduction in CDR, and that the CBN fragment of the $\overline{\lambda}\mu\tilde{\mu}$-calculus can model reduction in CDR. CDR is a Call by Push Value calculus (CBPV) in that it separates terms in computations and values, with their corresponding types. Some simulation results were already achieved by others for CBPV, but only up to equality for CBV; for CBN the results are rather weak.

In order to achieve a single-step reduction respecting mapping for the CBN $\lambda$-calculus, we allow forcing only for variables, thunking only for computations that are not forced variables, and change the nature of term substitution, and abolish the $U$-reduction rule. We will show that, by changing the standard interpretation, we can achieve a reduction respecting mapping for the CBV $\lambda$-calculus as well. Moreover, these changes make it possible to establish a strong relation between CDR and $\overline{\lambda}\mu\tilde{\mu}$, allowing to simulate CDR reduction in $\overline{\lambda}\mu\tilde{\mu}$, and preserving assignable types.

**keywords:** Call by Name, Call by Value, Semantics, Small step reduction, Type assignment, Soundness.

## Introduction

Both the 'call by name' (CBN) and 'call by value' (CBV) [19] reduction strategies of the $\lambda$-calculus [5, 2] are important paradigms in theory as well as in practice of computer science. The first is based on the concept that the execution of function application (the contraction of a $\beta$-redex) takes places 'as is', without consideration of the computational status of the argument. The second is similar, but the main distinction is that it forces the evaluation of the argument to a value before the redex under consideration gets contracted. CBN-reduction is normalising, so 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; it is an interesting question if a simple calculus can be found[1] in which both can be modelled coherently, preserving typeability, assignable types, denotational semantics, etc.

In [14, 16] Levy proposed the Call by Push Value (CBPV) calculus as answer to this question, as a subsuming paradigm for the CBN and CBV paradigms of the $\lambda$-calculus. Departing from earlier work by Moggi [17], Levy defines a calculus that distinguishes values from computations, not only syntactically, but also through assignable types. It introduces the concepts of *thunking* $\{\cdot\}$ (blocking a computation, turning it into a value) and *forcing* $\cdot!$ (used to unblock a thunked computation).

[1] Of course, both can be mapped into the $\lambda$-calculus itself.

An important characteristic of CBPV is that it is presented with a deterministic reduction relation, that can be thought of as encoded in the term itself, as reduction might be seen as being blocked and activated through additional syntax;[2] this strategy only allows some redexes occurring in terms to be contracted, and as will be argued in this paper, this basically follows a CBN-strategy. In particular, reduction in CBPV is not allowed for operands, *i.e.* in the right hand terms of applications; this implies that modelling CBV reduction is not straightforward, since that explicitly asks for the evaluation of operands. Levy defines CBN and CBV interpretations of the pure $\lambda$-calculus; as is usual, since their reduction relation is different, the CBN and CBV reduction strategies of the $\lambda$-calculus are interpreted differently into CBPV. He states preservation results for both reduction strategies; that this can be achieved in CBPV is rather remarkable, given that its reduction system is CBN in nature.

Approaches similar to CBPV were later explored in [7]; that paper presents the Bang calculus, which syntax can be mapped to that of CBPV, but allows for reduction to take place in all sub-terms, not just the computations, so representation of reductions of the CBN/CBV $\lambda$-calculus is not really an issue.

However, although defined exactly for this purpose, Levy's interpretations fall short in that they do not fully preserve the operational and denotational semantics. Under the CBV interpretation, only equality is preserved; the CBN interpretation is not a function, so the notion of semantics is very loose. For the latter, this is caused by a 'growth' of $\{\cdot\}!$ constructs created by reduction; under reduction, $(x!)\{\{M\}/x\}$ creates $\{M\}!$, also inside values where these cannot be contracted (see Ex. 2.17). As we will see through the proofs we supply for these properties in Sect. 2.1, these difficulties cannot be overcome in CBPV, so CBV and CBN reduction for the $\lambda$-calculus cannot be represented through reduction in CBPV.

So the question remains: is it possible to define a CBPV-like calculus in which we can faithfully represent CBV and CBN reduction for the $\lambda$-calculus? In this paper, we will answer this question positively, modifying CBPV slightly.

However, before coming to that, we need to address a discrepancy between the $\lambda$-calculus and CBPV. The discourse of Levy's papers on CBPV is that of terms, types and semantics, presented, as is perhaps more common in the context of programming languages, as derivations (called terms by Levy) for statements of the shape $\Gamma \vdash M : \mathsf{A}$; this means not only that terms are intrinsically typed, but also that terms and types cannot be separated, and reduction is in fact cut-elimination. Moreover, this way all terms are considered to be typeable, limiting the calculus and results. For the $\lambda$-calculus, it is more common to define terms and type assignment separately (so decidability of the latter becomes an issue), and reduction is a relation on (pure) terms, not depending on typeability.

It is perhaps surprising that Levy [16] nonetheless treats the witnesses of the derivations as terms separately, without mentioning their type; for example, large-step semantics is defined on terms only, ignoring that equality between terms can only soundly be defined *in a type*: rather than $M \Downarrow N$, as used by Levy, one should use $M \Downarrow_{\mathsf{B}} N$, so demand that there exists a $\Gamma$ such that both $\Gamma \vdash M : \mathsf{B}$ and $\Gamma \vdash N : \mathsf{B}$, and that the corresponding derivations for these judgements are related through cut elimination. Also, the notion $[V/x]$ as used in the definition of $\Downarrow$ in [16] should be defined on derivations (as in $(\Gamma, x{:}A \vdash_{\mathrm{C}} M : \mathsf{B}\ [(\Gamma \vdash_{\mathrm{V}} V : A)/x{:}A]$ or something similar), demanding that the type for $V$ and $x$ correspond. Fig. 6 of [16] defines an operational semantics through the abstract machine CK (see Def. 2.8), discarding the types and the necessity to define CK on type derivations as well. This leads to ambiguity and cannot be ignored.

[2] This is not true in reality, since the reduction steps do not check for the presence of this additional syntax, and reduction is actually, as always, controlled by evaluation contexts, that specify where redexes can be contracted. As can be seen in Def.. 2.2, for CBPV those are defined without even using thunking or forcing. In fact, these added constructs mainly aid type assignment.

In this paper we close this gap by extracting a pure, type-free term calculus, called CBPV as well, out of Levy's definition [14, 16], engineer a single-step reduction relation on those terms from the large-step semantics defined there, and define type assignment separately. Thereby it will be easier to link the $\lambda$-calculus and CBPV, without ignoring any aspect of terms. As a consequence, we can now deal with CBPV terms that are untypeable, like $\lambda x.x!\,x$, or non-terminating, like $(\lambda x.x!\,x)\{\lambda x.x!\,x\}$. In order to correctly represent the CBN/CBV $\lambda$-calculus, we will define a calculus of delayed reductions (CDR) as a variant of our CBPV, where we basically exclude terms of the shape $\{M\}!$, changing not only the syntax of terms, but also term-substitution. To validate our variant of CBPV, using this minor change, we will show that now it *is* possible to define interpretations of the CBN and the CBV-$\lambda$-calculus into CDR, and show that (single step) reduction is respected by (multistep) reduction, in Sect. 3.1 and 3.2, respectively, something which was not achieved for CBPV. For CBN this is achieved through changing the substitution, as explained above; for CBV, we enhance Levy's interpretation, which is not fine-grained enough for our purpose, returning $y := [\![N]\!]^{\lambda}_{\mathrm{v}}; [\![\lambda x.M]\!]^{\lambda}_{\mathrm{v}} y$ for $[\![(\lambda x.M)\,N]\!]^{\lambda}_{\mathrm{v}}$, rather than $x := [\![\lambda x.M]\!]^{\lambda}_{\mathrm{v}}; y := [\![N]\!]^{\lambda}_{\mathrm{v}}; x!\,y$ as does Levy.

[8] presents CBPV as a pure term calculus with type assignment, as we do here, but in that paper the discrepancy we indicated above is not mentioned. Forster et al. [8] basically repeat Levy's results by using Levy's definitions (so do not modify term substitution) and show that the interpretations are "correct w.r.t. small-step semantics, and using eager lets to eliminate administrative redices [sic]" [8]. Thereby their results do not corresponds to ours; the same observations on the shortcomings of Levy's representation results we made above hold for [8] as well; in fact, it is even difficult to compare [8]'s results with our own, given the differences in the CBN/CBV calculi that get interpreted into CBPV.

CBPV is not the only calculus that achieves the embodiment of the duality of CBN and CBV. Herbelin and Curien [11, 4] defined the calculus $\overline{\lambda}\mu\tilde{\mu}$ that represents proofs and cut-elimination of a variant Gentzen's Sequent Calculus for Classical Logic [9] with focus, which can be seen as a generalisation and extension of Parigot's $\lambda\mu$-calculus [18], and has been shown to represent an abstract machine which models both these paradigms successfully. In particular, the interpretations of the CBN and CBV $\lambda$-calculus into $\overline{\lambda}\mu\tilde{\mu}$ preserve the reductions.

Moreover, the abstract machine, CK, defined in [14] to give an operational semantics for CBPV, very closely corresponds to the workings of these interpretations into $\overline{\lambda}\mu\tilde{\mu}$. This observation led us to investigate the link between these two calculi - $\overline{\lambda}\mu\tilde{\mu}$ and CBPV - and here we will give an interpretation of CDR into $\overline{\lambda}\mu\tilde{\mu}$, inspired by the behaviour of CBPV in the abstract machine CK [15]. Using the motivation for $\overline{\lambda}\mu\tilde{\mu}$ as an abstract machine to run the $\lambda$-calculus in, we attempted to use a similar technique for translating CBPV. However, the halting and continuing of computations - through 'thunking' and 'forcing' - cannot be modelled by the traditional $\overline{\lambda}\mu\tilde{\mu}$ syntax, which gives another motivation to restrict the syntax and notion of reduction as we do here for CDR: especially the reduction rule $\{M\}! \to M$ cannot be represented in $\overline{\lambda}\mu\tilde{\mu}$. [6] studies the relation between CBPV and $\Lambda_{\mathrm{LLT}}$, which is a fully polarized typed calculus, inspired by $\overline{\lambda}\mu\tilde{\mu}$; it does not present embeddings of the CBN/CBV $\lambda$-calculus, which would be similar those defined by Levy, so does not solve the problems we address here.

However, when defining an interpretation of CDR-terms into $\overline{\lambda}\mu\tilde{\mu}$, as well as one for our variant of Levy's CK into $\overline{\lambda}\mu\tilde{\mu}$, we can show that both fully preserve reduction, and that the first also preserves typeability. In fact, reduction in CDR is fully modelled in CBN-$\overline{\lambda}\mu\tilde{\mu}$, emphasising again that CBPV is, in fact, a call-by-name calculus.

### Note for the reader

In order to explain the essence of our approach and solutions, in this paper we will revisit some results shown in [14, 16] and [8], so that we can show the strength of the results shown

there, and point out the (relative) weaknesses. As a result, a large part of this paper seems to present known results; however, this is not the case. As explained above, most of the results of [14, 16] are stated for a fundamentally different calculus, and mostly not shown or even defined in full detail, which here we will provide. Also the notion of the CBV $\lambda$-calculus differs significantly. Those details are important to understand why certain results in those other papers are not as strong as perhaps one would like, and serve to highlight the advantages of our approach; proofs can now be compared in detail, and exactly why our approach delivers the results is put into evidence.

### Outline of this paper

We start in Section 1 by giving a quick overview of the $\lambda$-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 2.1 we will revisit some results on the interpretation of CBN-$\lambda$ and CBV-$\lambda$ into CBPV, as well as those presented in *$\lambda$-val* [8] and the Bang calculus.

To address the shortcomings we observe in Section 2.1, in Section 3 we will present CDR, 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-$\lambda$ and CBV-$\lambda$ into CDR 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 5.

In Section 4 we will give a short overview of Herbelin's calculus $\overline{\lambda}\mu\tilde{\mu}$. This is followed in Section 5 by the definition of an interpretation of CDR-terms into $\overline{\lambda}\mu\tilde{\mu}$, and one for the abstract machine CK into $\overline{\lambda}\mu\tilde{\mu}$; we show that both fully preserve reduction, and that the first also preserves typeability.

## 1 The $\lambda$-calculus

We assume the reader to be familiar with the $\lambda$-calculus [2]; we just recall the definition of $\lambda$-terms and notions of reduction.

**Definition 1.1** (LAMBDA TERMS, CBN AND CBV REDUCTION) *i*) *$\lambda$-terms* are defined by the grammar:

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

In $MN$, we say that $M$ is *in function position*, and $N$ is *an operand*. We will write $M \in \lambda$ when $M$ is a $\lambda$-term.

*ii*) (One-step) $\beta$-reduction is defined using the $\beta$-rule

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

where $\{N/x\}$ stands for the implicit substitution[3] that is to take place immediately and silently, and evaluation contexts that are defined as terms with a single hole by:

$$\mathsf{C} ::= \lceil \rfloor^{4} \mid \mathsf{C}M \mid M\mathsf{C} \mid \lambda x.\mathsf{C}$$

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

[4] 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.

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

$$\mathsf{C}\lceil(\lambda x.M)\,N\rfloor \to_\beta \mathsf{C}\lceil M\{N/x\}\rfloor$$

for any evaluation context. We write $\to_\beta^*$ for the reflexive and 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:

$$\mathsf{C_N} ::= \lceil\rfloor \mid \mathsf{C_N}\,M$$

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

$$\mathsf{C_N}\lceil(\lambda x.M)\,N\rfloor \to_{\mathrm{N}} \mathsf{C_N}\lceil M\{N/x\}\rfloor$$

(also known as *lazy* reduction).

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

$$\mathsf{C_V} ::= \lceil\rfloor \mid \mathsf{C_V}\,M \mid V\,\mathsf{C_V}$$

*Call-by-value (*CBV*) reduction*, $\to_{\mathrm{V}}$, is defined through:

$$\mathsf{C_V}\lceil(\lambda x.M)\,V\rfloor \to_{\mathrm{V}} \mathsf{C_V}\lceil M\{V/x\}\rfloor$$

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 $\lambda$-term such that $M_1$ is not an application, then either $M_1 \equiv \lambda x.N$, or $M_1 \equiv y$. If $M_1 \equiv \lambda x.N$, then CBN-reduction on $M$ will contract the redex $(\lambda x.N)\,M_2$ to obtain the term $N\{M_2/x\}\,M_3\cdots M_n$, so without touching $M_2$. On the other hand, CBV-reduction will then 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 $N\{V_2/x\}\,M_3\cdots M_n$.

It is well known that reduction using $\to_\beta$ is confluent [3]; this is not a trivially achieved property, as $\to_\beta$ is not deterministic: by the definition of evaluation contexts, nested redexes can occur, as well as parallel redexes. For example, using $\to_\beta$, the three redexes in

$$(\lambda x.(\lambda y.M)\,N)\,((\lambda z.P)\,Q)$$

can all be contracted. Using $\to_{\mathrm{N}}$, only the '$x$'-redex can be contracted, and using $\to_{\mathrm{V}}$, only the '$z$'-redex. $\to_{\mathrm{N}}$ and $\to_{\mathrm{V}}$ are, in fact, reduction *strategies*: only ever one redex in a term can be contracted, so then reduction is deterministic.

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

**Definition 1.3** (CURRY TYPE ASSIGNMENT FOR THE $\lambda$-CALCULUS) *i*) Let $\varphi$ 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* $\Gamma$ 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* '$\vdash_\lambda$' is defined by the following inference system:

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

Type assignment in this system is decidable, and it enjoys a Curry-Howard correspondence with implicative intuitionistic logic.

## 2 Call By Push Value

Call by Push Value, presented in [14], is a calculus designed to make the execution order of a $\lambda$-based term explicit, by presenting a deterministic reduction system, but one that is set up to express both the CBN and CBV $\lambda$-calculus. CBPV considers *computations* and *values*; the slogan from [14] is

"*a value is, a computation does*"

and this is expressed through assignable types and the reduction relation, where only computations reduce to each other,[5] and values are used as parameters. However, the notion of value in CBPV is quite different from that used in the $\lambda$-calculus (see Def. 1.1).

We believe that this difference creates confusion when studying the relation between CBPV and the CBV $\lambda$-calculus; using the same terminology for two very different categories of terms suggest a correspondence between conceptually very different calculi. In both formalisms, values are the terms that get passed around and are substituted for term variables, but what these are differs significantly. In the CBV $\lambda$-calculus, values are essentially abstractions, whereas in CBPV, a computation $M$ becomes a value once it is 'thunked' $\{M\}$, effectively blocking reductions inside $M$; a value $V$ can be promoted to be a computation when it is 'forced' $V!$, unblocking the reductions. Since computations are the only terms that can be reduced, and run to computations, the result of reduction is never a value in the sense of CBPV. In fact, the only way a computation can be considered to return a value is when the outcome is of the shape $\mathit{ret}\, V$; the wrapper *ret* gets removed to produce a value during the contraction of rule $(F)$. A value $\{M\}$ can be transformed into a computation through the reduction process, by substituting it for a variable $x$ in $x!$, creating $\{M\}!$, which contracts to $M$, unblocking the computation.

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, $\lambda$-abstractions, applications, sequencing, and returning, and values are variables and thunked computations. Sequencing and returning play a major role in the encoding of CBV $\lambda$, 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, as we will argue below, its name notwithstanding, reduction in CBPV is mainly 'by name' (or lazy) in nature; this is reflected in the ease with which CBN $\lambda$ can be encoded. Its type system has a Curry-Howard-style correspondence with intuitionistic linear logic [20], hence only intuitionistic calculi can be translated into CBPV.

As mentioned in the introduction, in investigating the relation between reduction in the $\lambda$-calculus, defined on terms, and CBPV, defined through cut-elimination, we are confronted with a paradigm problem. In order to better compare the CBN/CBV $\lambda$-calculus and CBPV, here we will deviate from Levy's 'derivation' approach, and treat CBPV as a pure $\lambda$-calculus: we will define syntax and type assignment separately, and define a one-step reduction relation on terms.[6] This approach has also been used in [8].[7] This will enable us to interpret the untyped CBN/CBV $\lambda$-calculus without hindrances, but more importantly let us deal with non-terminating or untypeable terms as well. We will use the syntax from [8] but for reasons of simplicity consider pure, functional CBPV without effects and do not the consider sum and

[5] This is not the case in [7], see Sect. 2.5.

[6] Therefore, formally, none of the results we show for our variant of CBPV are applicable to Levy's calculus.

[7] It might be better to find a new name for this 'pure term' variant, but since [8] uses CBPV, to not introduce too much new nomenclature, so do we.

product types[8] that were included in the original definition of CBPV.

**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:

$$\begin{array}{ll} V, W ::= x \mid \{M\} & \textit{(values)} \\ M, N ::= V! \mid \lambda x.M \mid MV \mid \mathit{ret}\, V \mid x := M; N \,^{9} & \textit{(computations)} \end{array}$$

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 $M$. We will write $M \in \text{CBPV}$ when $M$ is a CBPV term.

As usual, we read $\{\cdot\}$ as 'thunk', which represents the blocking of a computation by boxing it inside a value, and $\cdot!$ as 'force', which pushes a value into becoming a computation.

We based a one-step reduction relation on the big-step semantics in Fig. 4 in [16].

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

$$\begin{array}{lcll} (C): & (\lambda x.M)\, V \to_{\mathrm{P}} M\{V/x\} & (\textit{contract})\,^{10} \\ (U): & \{M\}! \to_{\mathrm{P}} M & (\textit{unblock}) \\ (F): & x := \mathit{ret}\, V; M \to_{\mathrm{P}} M\{V/x\} & (\textit{force}) \end{array}$$

where term substitution $M\{V/x\}$ is defined as usual.

The CBPV-evaluation contexts are defined through: $\mathsf{C} ::= \lceil \rfloor \mid \mathsf{C}V \mid x := \mathsf{C}; M$ and CBPV-reduction is defined through:

$$\mathsf{C}\lceil M \rfloor \to_{\mathrm{P}} \mathsf{C}\lceil N \rfloor$$

whenever $M \to_{\mathrm{P}} N$ through either rule $(C)$, $(U)$, or $(F)$ and $\mathsf{C}$ is a CBPV-evaluation context.

Notice that reduction is deterministic: since the definition of evaluation contexts lack the cases $M\mathsf{C}$, $\lambda x.\mathsf{C}$ and $x := M; \mathsf{C}$, no nested or parallel redexes can occur.

The constructs $\mathit{ret}\, V$ and $x := M; N$ are not generated by reduction; in our version of CBPV they are present mainly for the encoding of the CBV $\lambda$-calculus (see Def. 2.11).

Levy [16] only considers semantics for closed terms, so $V$ is closed in $x := \mathit{ret}\, V; M$, and $M$ has only one free variable, $x$. Here we consider arbitrary terms, so $V$ can have free variables, and even start with one (be of the shape $\{y!\, V_1 \cdots V_n\}$).[11]

*Example 2.3* We have $(\lambda x.x!)\{y!\, y\} \to_{\mathrm{P}} \{y!\, y\}! \to_{\mathrm{P}} y!\, y$ whereas the corresponding $\lambda$-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:

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

[8] 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.

[9] Many different notations have been used for CBPV in the past. [16] writes application $MV$ 'operand first' as $V$'$M$ in and past papers used *let* $x = M$ *in* $N$ or even *let* $x \leftarrow M$ *in* $N$ for our $x := M; N$.

[10] This rule is called $(\to)$ in [16].

[11] In general, reduction rule $(F)$ does not require the thunked term in $N$ in $x := \mathit{ret}\{N\}; M$ to be in normal form; any computation term can be thunked and passed on, even $N = (\lambda y.P)\, Q$.

**Definition 2.4** (CBPV TYPE ASSIGNMENT) *i*) The types[12] for CBPV terms are defined by:

$$A, B ::= \varphi \mid U\mathsf{A} \qquad (\textit{value types})$$
$$\mathsf{A}, \mathsf{B} ::= A \to \mathsf{B} \mid FA \qquad (\textit{computation types})$$

We will write $UFA$ for $U(FA)$ and $F U\mathsf{A}$ for $F(U\mathsf{A})$.

*ii*) Contexts are defined as in Def. 1.3, but mapping term variables to value types.

*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.

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

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

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

*Example 2.5* There are various ways of typing 'identity' in CBPV; notice that the term $\lambda x.x$ is not a CBPV-term, since by the grammar, we can only abstract over computations. Since only value types are allowed for term variables, the statement $x{:}A$ as occurs in the context stands for $x{:}\varphi$ or $x{:}U\mathsf{A}$. This implies that we have two options for identity:

$$\dfrac{\dfrac{\dfrac{}{x{:}\varphi \vdash_{\mathrm{v}} x : \varphi}(\textit{axiom})}{x{:}\varphi \vdash_{\mathrm{c}} \mathit{ret}\, x : F\varphi}(\textit{ret})}{\vdash_{\mathrm{c}} \lambda x.\mathit{ret}\, x : \varphi \to F\varphi}(\textit{abstr}) \qquad \dfrac{\dfrac{\dfrac{}{x{:}UF\varphi \vdash_{\mathrm{v}} x : UF\varphi}(\textit{axiom})}{x{:}UF\varphi \vdash_{\mathrm{c}} x! : F\varphi}(\textit{force})}{\vdash_{\mathrm{c}} \lambda x.x! : UF\varphi \to F\varphi}(\textit{abstr})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x{:}U(U\mathsf{A}\to\mathsf{A}) \vdash_{\mathrm{v}} x : U(U\mathsf{A}\to\mathsf{A})}(\textit{axiom})}{x{:}U(U\mathsf{A}\to\mathsf{A}) \vdash_{\mathrm{c}} x! : U\mathsf{A}\to\mathsf{A}}(\textit{force})}{\vdash_{\mathrm{c}} \lambda x.x! : U(U\mathsf{A}\to\mathsf{A}) \to (U\mathsf{A}\to\mathsf{A})}(\textit{abstr}) \qquad \dfrac{\dfrac{\dfrac{}{x{:}U\mathsf{A} \vdash_{\mathrm{v}} x : U\mathsf{A}}(\textit{axiom})}{x{:}U\mathsf{A} \vdash_{\mathrm{c}} x! : \mathsf{A}}(\textit{force})}{\vdash_{\mathrm{c}} \lambda x.x! : U\mathsf{A}\to\mathsf{A}}(\textit{abstr})}{\vdash_{\mathrm{c}} \{\lambda x.x!\} : U(U\mathsf{A}\to\mathsf{A})}(\textit{thunk})}{\vdash_{\mathrm{c}} (\lambda x.x!)\{\lambda x.x!\} : U\mathsf{A}\to\mathsf{A}}(\textit{appl})$$

so 'identity' either takes in a value and returns it, or takes in a thunked argument and forces it. Likewise, taking $\Gamma = x{:}U(\varphi\to\mathsf{B}), y{:}\varphi$, and $\Gamma' = x{:}U(U\mathsf{A}\to\mathsf{B}), y{:}U\mathsf{A}$ we can derive:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash_{\mathrm{v}} x : U(\varphi\to\mathsf{B})}(\textit{axiom})}{\Gamma \vdash_{\mathrm{c}} x! : \varphi\to\mathsf{B}}(\textit{force}) \quad \dfrac{}{\Gamma \vdash_{\mathrm{v}} y : \varphi}(\textit{axiom})}{\Gamma \vdash_{\mathrm{c}} x!y : \mathsf{B}}(\textit{appl})}{x{:}U(\varphi\to\mathsf{B}) \vdash_{\mathrm{c}} \lambda y.x!y : \varphi\to\mathsf{B}}(\textit{abstr})}{\vdash_{\mathrm{c}} \lambda xy.x!y : U(\varphi\to\mathsf{B})\to\varphi\to\mathsf{B}}(\textit{abstr}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma' \vdash_{\mathrm{v}} x : U(U\mathsf{A}\to\mathsf{B})}(\textit{axiom})}{\Gamma' \vdash_{\mathrm{c}} x! : U\mathsf{A}\to\mathsf{B}}(\textit{force}) \quad \dfrac{}{\Gamma' \vdash_{\mathrm{v}} y : U\mathsf{A}}(\textit{axiom})}{\Gamma' \vdash_{\mathrm{c}} x!y : \mathsf{B}}(\textit{appl})}{x{:}U(U\mathsf{A}\to\mathsf{B}) \vdash_{\mathrm{c}} \lambda y.x!y : U\mathsf{A}\to\mathsf{B}}(\textit{abstr})}{\vdash_{\mathrm{c}} \lambda xy.x!y : U(U\mathsf{A}\to\mathsf{B})\to U\mathsf{A}\to\mathsf{B}}(\textit{abstr})$$

We can observe that the type for $\lambda xy.x!y$, the equivalent for $\lambda xy.xy$ in CBPV (which models application) is *not* $(A\to\mathsf{B})\to A\to\mathsf{B}$, as the rule $(\textit{appl})$ would suggest.

We can even derive the following (where $\Gamma = x{:}U(U\mathsf{A}\to\mathsf{B}), y{:}U\mathsf{A}$):

[12] We choose to use a different font for computation types for reasons of readability, rather than underlining as in [14], or $C, D$ as in [8].

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash_{\mathrm{v}} x : U(U\mathsf{A}\to\mathsf{B})}\,(axiom)}{\Gamma \vdash_{\mathrm{c}} x! : U\mathsf{A}\to\mathsf{B}}\,(force)}{\Gamma \vdash_{\mathrm{v}} \{x!\} : U(U\mathsf{A}\to\mathsf{B})}\,(thunk)}{\Gamma \vdash_{\mathrm{c}} \{x!\}! : U\mathsf{A}\to\mathsf{B}}\,(force) \qquad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash_{\mathrm{v}} y : U\mathsf{A}}\,(axiom)}{\Gamma \vdash_{\mathrm{c}} y! : \mathsf{A}}\,(force)}{\Gamma \vdash_{\mathrm{v}} \{y!\} : U\mathsf{A}}\,(thunk)}{\Gamma \vdash_{\mathrm{c}} \{x!\}!\{y!\} : \mathsf{B}}\,(appl)}{x{:}U(U\mathsf{A}\to\mathsf{B}) \vdash_{\mathrm{c}} \lambda y.\{x!\}!\{y!\} : U\mathsf{A}\to\mathsf{B}}\,(abstr)}{\vdash_{\mathrm{c}} \lambda xy.\{x!\}!\{y!\} : U(U\mathsf{A}\to\mathsf{B})\to U\mathsf{A}\to\mathsf{B}}\,(abstr)$$

indicating that CBPV allows for some superfluous thunking and forcing; it seems that perhaps the syntax of CBPV-terms could be restricted to not permit thunking of forced terms. We will further explore this idea in this paper in Section 3.

This polarised type assignment highlights the duality of CBPV, using initial and terminal objects; it assigns value types to values, and computation types to computations. It does that by dividing Moggi's type constructor $T$ [17] into two type constructors $U$ and $\mathsf{F}$, and enriches Moggi's $\lambda_{ml}$ with thunking and forcing that allow the conversion form computation to values and vice versa; the intention is that a computation of type $\mathsf{F}A$ *produces* a value of type $A$, and a value of type $U\mathsf{B}$ is a *thunk* of a computation of type $\mathsf{B}$. Notice that $\mathsf{F}\varphi$ is the basic computational type, where $\varphi$ is the basic value type.

In the present context, the role of $ret\,\cdot$ is rather limited, and we will see that it plays no part in the interpretation of the CBN $\lambda$-calculus (Def. 2.16 and 3.2). However, when adding term constants, like numbers, to CBPV, the role of $ret\,\cdot$ becomes more evident: then all numbers are values of type $num$, and, for example, successor $Succ$ is typed $Succ : num \to \mathsf{F}num$, so $Succ\,0 : \mathsf{F}num$. Note that results of computations of the shape $ret\,V$ cannot be passed to functions without being thunked first, and can only be passed on using the assignment construct.

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

*Lemma 2.6* *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* $\Gamma \vdash_{\mathrm{c}} M\{V'/x\} : \mathsf{B}$.

*Proof:* By simultaneous induction on the definition of '$\cdot\{V/x\}$'. □

**Theorem 2.7** (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 by rules $(appl)$ and $(abstr)$ 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 Lem. 2.6.
$(U)$: Then $M = \{M'\}!$, and we have $\Gamma \vdash_{\mathrm{c}} M' : \mathsf{B}$ by rules $(thunk)$ and $(force)$.
$(F)$: Then $M = x := ret\,V; M'$ 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 Lem. 2.6. □

In the same vein as Krivine's machine for the $\lambda$-calculus [13], in [16] Levy defines a small-step operational semantics of CBPV through a stack machine CK, which uses configurations $\langle M \,|\, S\rangle$ 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 that evaluation of $M$ takes place. We will present a variant of that machine here, adapted to our restriction of CBPV, so without effects or sum and product types.

**Definition 2.8** (CK-MACHINE [16]) *i) Evaluation stacks* are defined over CBPV terms and values through: $S ::= \epsilon \mid V : S \mid x := [\,]; M : S$.

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

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

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

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

Notice that in the last step of part (*ii*), there is no interaction with the stack.

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

**Theorem 2.9** *If $M \to_{\text{P}} N$, then $\langle M \mid S\rangle =_{\text{CK}} \langle N \mid S\rangle$.*

*Proof:* $(\lambda x.M)\, V \to_{\text{P}} M\{V/x\}$: $\langle (\lambda x.M)\, V \mid S\rangle \to_{\text{CK}} \langle \lambda x.M \mid V : S\rangle \to_{\text{CK}} \langle M\{V/x\} \mid S\rangle$

$\{M\}! \to_{\text{P}} M$: $\langle \{M\}! \mid S\rangle \to_{\text{CK}} \langle M \mid S\rangle$

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

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

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

Notice that the result is stated in terms of equality, rather than reduction, because of the structure of the last two cases in the proof; see also the comment below. With this result, we immediately have that CBPV-reduction is preserved by the interpretation $[\![\cdot]\!]^{\text{CK}}$.

*Corollary 2.10* *If $M \to_{\text{P}}^{*} N$, then $[\![M]\!]^{\text{CK}} =_{\text{CK}} [\![N]\!]^{\text{CK}}$.*

The last property is stated in [16] (adapted here to our notation) as 'For any closed computation $M$, we have $M \Downarrow T$ iff $\langle M \mid \epsilon\rangle \to_{\text{CK}}^{*} \langle T \mid \epsilon\rangle$' (note that this does not imply that small-step reduction is respected); our result generalises this to single-step reduction, but at the cost of using $=_{\text{CK}}$. However, this does not imply that reduction is respected, since we have:

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

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

We will see this problem come back in Thm. 2.13.

## 2.1 Interpreting the CBN/CBV $\lambda$-calculus in CBPV

There are two different ways to interpret the $\lambda$-calculus in CBPV, by either modelling CBN or CBV reduction, and the syntax of CBPV allows to explicitly encode either strategy; it is not possible to model full $\beta$-reduction, since the evaluation contexts $\lambda x.\mathsf{C}$ and $M\mathsf{C}$ are missing in CBPV. Levy presents in [16] what he calls Fine-Grain CBV, a typed $\lambda$-calculus extended with two `let` constructs and a conditional construct (presented as derivations); a variant of that calculus is considered in [8], but there with product and sum, and no longer considering terms of CBPV to be derivations. Since in this paper we are mainly interested in the relation between the pure $\lambda$-calculus, CBPV, and $\overline{\lambda}\mu\tilde{\mu}$, all untyped, we will concentrate on the pure functional component of those calculi and not use Levy's approach.

## 2.2 A CBV-interpretation of $\Lambda$ in CBPV

Rather than following the path of [16, 8] where extended calculi are studied, we will here focus on interpreting the pure CBV $\lambda$-calculus (Def. 1.1) in CBPV, so start from the thought that CBV is a *reduction strategy* for the $\lambda$-calculus, not a different calculus that requires changing the syntax. Since the calculi involved are different, the results we present in this section are not those of [16, 8], so all proofs are new.

**Definition 2.11** ([16]) Levy's CBV-interpretation $\mapsto_{\mathrm{v}}$ between $\lambda$-terms and CBPV-terms is defined through:

$$\frac{}{x \mapsto_{\mathrm{v}} ret\, x} \qquad \frac{M \mapsto_{\mathrm{v}} P}{\lambda x.M \mapsto_{\mathrm{v}} ret\{\lambda x.P\}} \qquad \frac{M \mapsto_{\mathrm{v}} P \qquad N \mapsto_{\mathrm{v}} Q}{MN \mapsto_{\mathrm{v}} x := P; y := Q; x!y}$$

Notice the use of sequencing: it has the effect that, when interpreting an application $MN$, it first runs (the interpretation of) $M$, followed by $N$, and then applying the resulting terms, as is intended under CBV. This reorganisation of terms is necessary because the reduction relation on CBPV is CBN, *not* CBV in the sense of the $\lambda$-calculus. Since $\mapsto_{\mathrm{v}}$ is a function, for notational convenience in proofs, we will write $[\![M]\!]^{\mathrm{L}}_{\mathrm{v}}$ for $P$ whenever $M \mapsto_{\mathrm{v}} P$.

We can show that this interpretation preserves reduction (through equality) and assignable types; we first show that it respects term-substitution.

*Lemma 2.12* (SUBSTITUTION LEMMA FOR $[\![\cdot]\!]^{\mathrm{L}}_{\mathrm{v}}$) *i)* $[\![M]\!]^{\mathrm{L}}_{\mathrm{v}}\{w/z\} = [\![M\{w/z\}]\!]^{\mathrm{L}}_{\mathrm{v}}$.
*ii)* $[\![M]\!]^{\mathrm{L}}_{\mathrm{v}}\{\{\lambda w.[\![R]\!]^{\mathrm{L}}_{\mathrm{v}}\}/z\} = [\![M\{\lambda w.R/z\}]\!]^{\mathrm{L}}_{\mathrm{v}}$.

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

$M \equiv u$ *and* $u \neq z$: $[\![u]\!]^{\mathrm{L}}_{\mathrm{v}}\{\{\lambda w.[\![R]\!]^{\mathrm{L}}_{\mathrm{v}}\}/z\} = (ret\, u)\{\{\lambda w.[\![R]\!]^{\mathrm{L}}_{\mathrm{v}}\}/z\} = ret\, u \stackrel{\Delta}{=} [\![u]\!]^{\mathrm{L}}_{\mathrm{v}} = [\![u\{\lambda w.R/z\}]\!]^{\mathrm{L}}_{\mathrm{v}}$
The other cases follow by induction. □

With this result we can show that the interpretation $[\![\cdot]\!]^{\mathrm{L}}_{\mathrm{v}}$ respects reduction up to equality.

**Theorem 2.13** *If* $M \to_{\mathrm{v}} N$, *then* $[\![M]\!]^{\mathrm{L}}_{\mathrm{v}} =_{\mathrm{P}} [\![N]\!]^{\mathrm{L}}_{\mathrm{v}}$.

*Proof:* $(\lambda z.M)\, w \to_{\mathrm{v}} M\{w/z\}$: $[\![(\lambda z.M)\, w]\!]^{\mathrm{L}}_{\mathrm{v}} \stackrel{\Delta}{=} x := ret\{\lambda z.[\![M]\!]^{\mathrm{L}}_{\mathrm{v}}\}; y := ret\, w; x!y \to_{\mathrm{P}} y := ret\, w; \{\lambda z.[\![M]\!]^{\mathrm{L}}_{\mathrm{v}}\}!y \to_{\mathrm{P}} \{\lambda z.[\![M]\!]^{\mathrm{L}}_{\mathrm{v}}\}!w \to_{\mathrm{P}} (\lambda z.[\![M]\!]^{\mathrm{L}}_{\mathrm{v}})\, w \to_{\mathrm{P}} [\![M]\!]^{\mathrm{L}}_{\mathrm{v}}\{w/z\} = (2.12) [\![M\{w/z\}]\!]^{\mathrm{L}}_{\mathrm{v}}$

$(\lambda z.M)\,(\lambda w.R) \to_{\mathrm{v}} M\{\lambda w.R/z\}$: $[\![(\lambda z.M)\,(\lambda w.R)]\!]^{\mathrm{L}}_{\mathrm{v}} \stackrel{\Delta}{=} x := ret\{\lambda z.[\![M]\!]^{\mathrm{L}}_{\mathrm{v}}\}; y := ret\{\lambda w.[\![R]\!]^{\mathrm{L}}_{\mathrm{v}}\}; x!y \to_{\mathrm{P}} y := ret\{\lambda w.[\![R]\!]^{\mathrm{L}}_{\mathrm{v}}\}; \{\lambda z.[\![M]\!]^{\mathrm{L}}_{\mathrm{v}}\}!y \to_{\mathrm{P}} \{\lambda z.[\![M]\!]^{\mathrm{L}}_{\mathrm{v}}\}!\{\lambda w.[\![R]\!]^{\mathrm{L}}_{\mathrm{v}}\} \to_{\mathrm{P}} (\lambda z.[\![M]\!]^{\mathrm{L}}_{\mathrm{v}})\{\lambda w.[\![R]\!]^{\mathrm{L}}_{\mathrm{v}}\} \to_{\mathrm{P}} [\![M]\!]^{\mathrm{L}}_{\mathrm{v}}\{\{\lambda w.[\![R]\!]^{\mathrm{L}}_{\mathrm{v}}\}/z\} =(2.12) [\![M\{\{\lambda w.R\}/z\}]\!]^{\mathrm{L}}_{\mathrm{v}}$

$M \to_{\mathrm{v}} N \Rightarrow MP \to_{\mathrm{v}} NP$: $[\![MP]\!]^{\mathrm{L}}_{\mathrm{v}} \stackrel{\Delta}{=} x := [\![M]\!]^{\mathrm{L}}_{\mathrm{v}}; y := [\![P]\!]^{\mathrm{L}}_{\mathrm{v}}; x!y =_{\mathrm{P}} (IH)$ $x := [\![N]\!]^{\mathrm{L}}_{\mathrm{v}}; y := [\![P]\!]^{\mathrm{L}}_{\mathrm{v}}; x!y \stackrel{\Delta}{=} [\![NP]\!]^{\mathrm{L}}_{\mathrm{v}}$

$M \to_{\mathrm{v}} N \Rightarrow zM \to_{\mathrm{v}} zN$: $[\![zM]\!]^{\mathrm{L}}_{\mathrm{v}} \stackrel{\Delta}{=} x := ret\, z; y := [\![M]\!]^{\mathrm{L}}_{\mathrm{v}}; x!y \to_{\mathrm{P}} y := [\![N]\!]^{\mathrm{L}}_{\mathrm{v}}; z!y$ and $[\![zN]\!]^{\mathrm{L}}_{\mathrm{v}} \stackrel{\Delta}{=} x := z; y := [\![N]\!]^{\mathrm{L}}_{\mathrm{v}}; x!y \to_{\mathrm{P}} y := [\![N]\!]^{\mathrm{L}}_{\mathrm{v}}; z!y$

$M \to_{\mathrm{v}} N \Rightarrow (\lambda w.R)\, M \to_{\mathrm{v}} (\lambda w.R)\, N$: $[\![(\lambda w.R)\, M]\!]^{\mathrm{L}}_{\mathrm{v}} \stackrel{\Delta}{=} x := ret\{\lambda w.[\![R]\!]^{\mathrm{L}}_{\mathrm{v}}\}; y := [\![M]\!]^{\mathrm{L}}_{\mathrm{v}}; x!y \to_{\mathrm{P}} y := [\![M]\!]^{\mathrm{L}}_{\mathrm{v}}; \{\lambda w.[\![R]\!]^{\mathrm{L}}_{\mathrm{v}}\}y =_{\mathrm{P}} (IH)$ $y := [\![N]\!]^{\mathrm{L}}_{\mathrm{v}}; \{\lambda w.[\![R]\!]^{\mathrm{L}}_{\mathrm{v}}\}y$ and

$[\![(\lambda w.R)\,N]\!]^{\mathrm{L}}_{\mathrm{V}} \stackrel{\Delta}{=} x := ret\,\{\lambda w.[\![R]\!]^{\mathrm{L}}_{\mathrm{V}}\}; y := [\![N]\!]^{\mathrm{L}}_{\mathrm{V}}; x!\,y \to_{\mathrm{P}} y := [\![N]\!]^{\mathrm{L}}_{\mathrm{V}}; \{\lambda w.[\![R]\!]^{\mathrm{L}}_{\mathrm{V}}\}\,y$ □

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.

The CBV $\lambda$-calculus and CBPV are related also on the level of types. We first define an interpretation of types.

**Definition 2.14** ([16]) The CBV-interpretation of Curry types into CBPV-types is defined through:

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

This interpretation is straightforwardly extended to contexts: $\overline{\Gamma} \stackrel{\Delta}{=} \{\, x{:}\overline{A} \mid x{:}A \in \Gamma \,\}$.

We can now show that the interpretations respect typeability and assignable types.

*Lemma 2.15* *If* $\Gamma \vdash_{\lambda} M : A$, *then* $\overline{\Gamma} \vdash_{\mathrm{C}} [\![M]\!]^{\mathrm{L}}_{\mathrm{V}} : \mathsf{F}\overline{A}$.

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

$(Ax)$: Then $M \equiv x$, and $\Gamma = \Gamma', x{:}A$; 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 $[\![x]\!]^{\mathrm{L}}_{\mathrm{V}} \stackrel{\Delta}{=} ret\,x$. □

### 2.3 A CBN-handling of $\Lambda$ in CBPV

In [16], Levy also deals with mapping CBN-reduction for the $\lambda$-calculus into CBPV, but this attempt is less successful than the one dealing with CBV; [8] basically repeats Levy's work and result, but for a pure term calculus. Under CBN-reduction in the $\lambda$-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 $\lambda$-terms into CBPV.

**Definition 2.16** (SIMULATION OF CBN [16, 8]) Levy's relation $\mapsto_{\mathrm{N}}$ between $\lambda$-terms and CBPV-terms is defined as:

$$\dfrac{}{x \mapsto_{\mathrm{N}} x!} \qquad \dfrac{M \mapsto_{\mathrm{N}} M'}{\lambda x.M \mapsto_{\mathrm{N}} \lambda x.M'} \qquad \dfrac{M \mapsto_{\mathrm{N}} M' \quad N \mapsto_{\mathrm{N}} N'}{MN \mapsto_{\mathrm{N}} M'\{N'\}} \qquad \dfrac{M \mapsto_{\mathrm{N}} M'}{M \mapsto_{\mathrm{N}} \{M'\}!}$$ [13]

Notice that '$x := N; M'$' and '$ret\,V$' are not used here, and that all variables are forced, even those that appear in value position (where they get thunked). Notice that we have

$$\{\, M \in \text{CBPV} \mid \lambda x.xx \mapsto_{\mathrm{N}} M \,\} = \{\, \lambda x.x!\{x!\}, \lambda x.x!\{\{x!\}!\}, \lambda x.x!\{\{\{x!\}!\}!\}, \ldots \}$$

and that these terms are not related through reduction, so $\mapsto_{\mathrm{N}}$ is not a (semantic) function.

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 $\{\cdot\}!$ around interpreted $\lambda$-terms, so for every $\lambda$-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.

[13] The last rule is missing from Fig. 3 in the Appendix of [8], but we assume this is in error.

[8] states that $\mapsto_{\mathrm{N}}$ is injective, but that seems to be an unnecessary claim for a relation that maps applications to applications, and abstractions to abstractions (modulo $\{\cdot\}!$), and is not even a function.

It is not possible to show that reduction is preserved under this relation.

*Example 2.17* Remark that $(\lambda x.xx)\,(\lambda x.xx) \to_{\mathrm{N}} (\lambda x.xx)\,(\lambda x.xx)$. We have:

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

and

$$\begin{aligned}(\lambda x.x!\{x!\})\{\lambda x.x!\{x!\}\} &\to_{\mathrm{P}} (C)\ \ x!\{x!\}\{\lambda x.x!\{x!\}/x\}\\ &=\ \lambda x.x!\{x!\}!\{\{\lambda x.x!\{x!\}\}!\}\\ &\to_{\mathrm{P}} (U)\ \ (\lambda x.x!\{x!\})\{\{\lambda x.x!\{x!\}\}!\}\\ &\to^*_{\mathrm{P}}\ (\lambda x.x!\{x!\})\{\{\{\lambda x.x!\{x!\}\}!\}!\}\\ &\to^*_{\mathrm{P}} \ldots\end{aligned}$$

Note that the $\{\cdot\}!$ redexes in the operand cannot be contracted. It will be clear that the term $(\lambda x.x!\{x!\})\{\lambda x.x!\{x!\}\}$ does not run to itself, and that the size of the term keeps increasing.

One could argue that it could have been better to reverse the above idea and add the rule $(U)$ without limitation to solve this problem; 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; this is essentially the solution chosen in [7] for the Bang calculus. However, we would need to allow this reduction step also inside values, breaking the slogan from [14], that "*a value is, a computation does*", which is probably why Levy does not consider this change. Below we will address this by eliminating forcing of thunked terms and rule $(U)$ altogether in CDR, without breaking the slogan.

It is possible to show that $\mapsto_{\mathrm{N}}$ respects type assignment; this result is very much like that of Thm. 3.16, and we therefore will not present it here.

Using this notion, [16, 8] show:

*Lemma 2.18* ([16, 8]) *i) (Forwards simulation) Let $M, N \in \lambda$. If $M \mapsto_{\mathrm{N}} Q$, and $M \to^*_{\mathrm{N}} N$, then there exists $R \in$ CBPV such that $N \mapsto_{\mathrm{N}} R$ and $Q \to^*_{\mathrm{P}} R$.*

*ii) (Backwards simulation) Let $M \in \lambda$. If $M \mapsto_{\mathrm{N}} Q$, and $Q \to^*_{\mathrm{P}} R$, then there exists $N \in \lambda$ such that $N \mapsto_{\mathrm{N}} R$ and $M \to^*_{\mathrm{N}} N$.*

This is a rather complicated solution for a problem that is easily fixed by making a different choice of grammar for CBPV. Since the problem is introduced by (implicit) substitution of a thunked term for a forced variable, 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 Sect. 3, where we present *essential* CBPV.

### 2.4 The results for $\lambda$-*val* [8]

In this section we will revisit the results shown for a variant of the $\lambda$-calculus presented in [16, 8] called the 'simply-typed fine-grained call-by-value $\lambda$-calculus' that distinguishes *terms* from *values* through the prefix construct *return* in [16], called *val* in [8]. It is called CBV throughout [8], but since it differs significantly from the original CBV $\lambda$-calculus, in order to distinguish

these we call it the $\lambda$-*val* calculus here. Reduction rules are not presented in [16]; to be able to show our results, we will define them below.

**Definition 2.19** (The $\lambda$-*val* calculus [8]) The terms of the $\lambda$-*val* calculus are defined using the grammar:

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

For type assignment, we use the types and variants of the inference rules from Def. 1.3, adding a rule that deals with *val*:

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

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

The name '*Call by Value*' is used in [8] for this calculus although it bears little resemblance to CBV-$\lambda$: notice that rule $(\textit{abstr})$ 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 [8] deal with the relation between CBN, CBV, and CBPV.

When mapping $\lambda$-*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 $\lambda$-calculus types into CBPV types.

**Definition 2.20** (Simulation of $\lambda$-*val* in CBPV [8]) Terms of the $\lambda$-*val* calculus are translated into those for CBPV through $\overline{\cdot}$:

$$\begin{aligned} \overline{x} &\triangleq x & \overline{\mathit{val}\, v} &\triangleq \mathit{ret}\, \overline{v} \\ \overline{\lambda x.s} &\triangleq \{\lambda x.\overline{s}\} & \overline{s\, t} &\triangleq x := \overline{s}; y := \overline{t}; x\, !\, y \end{aligned}$$

Types are translated using Levy's $\overline{\cdot}$ (Def. 2.14):

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

Notice that values are only allowed inside applications if preceded by the *val* keyword; this is the main difference between $\lambda$-*val* and the traditional CBV $\lambda$-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 [8].Lemma 2.3, which incorrectly restates the result shown in [14] as 'If $\Gamma \vdash_e s : A$, then $\overline{\Gamma} \vdash_{\mathrm{C}} \overline{s} : \overline{A}$ and analogously for values'.[14] We correct and prove the result here for our CBPV.

*Lemma 2.21* *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.

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

$(\textit{abstr})$: 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:

[14] The proofs in Coq provided online for [8] seem to avoid this error.

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

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 $\Gamma' = \overline{\Gamma}, x{:}\mathsf{U}(B \to \mathsf{A}), y{:}B$):

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

□

Reduction on $\lambda$-*val* is not formally defined in [8], but following common practice, we can assume it to be defined as follows.

**Definition 2.22** (Reduction on $\lambda$-*val*) *Evaluation contexts* for $\lambda$-*val* are defined through:

$$\mathsf{C}_{\mathrm{V}} ::= [\,] \mid \mathsf{C}_{\mathrm{V}}\, t \mid (val\, v)\, \mathsf{C}_{\mathrm{V}}$$

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

$$\mathsf{C}_{\mathrm{V}}\lceil (val\, \lambda x.s)\, (val\, v) \rfloor \to_v \mathsf{C}_{\mathrm{V}}\lceil s\{v/x\} \rfloor$$

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

*Example 2.23* We consider the reduction of $(val\, \lambda a.(val\, a)\, (val\, a))\, (val\, \lambda a.(val\, a)\, (val\, a))$ (which is the $\lambda$-*val* equivalent of the $\lambda$-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:

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

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 Def. 2.11).

Regardless of the missing formal definition, the authors claim that their 'translation is correct w.r.t. small-step semantics' [8]. We make the following observation:

*Example 2.24* 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:

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

and

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

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.[15]

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 [8] there is no real motivation given for the departure from the pure $\lambda$-calculus with CBV-reduction by adding the keyword *val*. In fact, all results shown in [8] were already claimed in [16], but using (an extension of) the pure $\lambda$-calculus with CBV-reduction. In particular, the problem of Thm. 2.13, that reduction is only respected up to equality, is still there.

## 2.5 The simulation results for the Bang calculus

Approaches similar to CBPV were later explored in [7], but from the perspective of linear logic [10]; it presents the Bang calculus, which corresponds to return/sequence-free CBPV, so is not based on $\lambda_{ml}$. It uses the syntax:[16]

$$\begin{array}{ll}
V, W ::= x \mid \{M\} & (values) \\
M, N ::= V \mid \lambda x.M \mid MN \mid M! & (terms)
\end{array}$$

[15] It could of course be that our definition of reduction in Def. 2.22 is not the one the authors of [8] intended, but that seems unlikely.

[16] Adapted here to CBPV notation; using [7]'s would create too much confusion, since it uses *!* for thunking.

on which it defines (weak) reduction rules, corresponding to rules $(C)$ and $(U)$ of CBPV (see Def. 2.1), that are allowed to contract in the (weak) evaluation contexts,

$$\mathsf{C} ::= \lceil\rfloor \mid \lambda x.\mathsf{C} \mid \mathsf{C}N \mid M\mathsf{C} \mid \mathsf{C}! \mid \{\mathsf{C}\}$$
$$\mathsf{W} ::= \lceil\rfloor \mid \lambda x.\mathsf{W} \mid \mathsf{W}N \mid M\mathsf{W} \mid \mathsf{W}!$$

essentially permitting contracting of *all redexes*, but for those occurring in thunked terms in weak contexts for weak reduction. Thereby reduction in the Bang calculus allows for the evaluation of operands until they become values, and mapping the CBV $\lambda$-calculus is rather straightforward. Moreover, since reduction is essentially free, even allowed under abstractions, $U$-redexes can be contracted anywhere, and modelling CBN is straightforward as well.

In particular, [7] presents encodings of both the CBN and CBV $\lambda$-calculus through:

$$\begin{array}{rcl@{\qquad}rcl} x^{\mathrm{cbn}} & \triangleq & x! & x^{\mathrm{cbv}} & \triangleq & x \\ (\lambda x.M)^{\mathrm{cbn}} & \triangleq & \lambda x.M^{\mathrm{cbn}} & (\lambda x.M)^{\mathrm{cbv}} & \triangleq & \{\lambda x.M^{\mathrm{cbv}}\} \\ (MN)^{\mathrm{cbn}} & \triangleq & M^{\mathrm{cbn}}\{N^{\mathrm{cbn}}\} & (MN)^{\mathrm{cbv}} & \triangleq & M^{\mathrm{cbv}}!\,N^{\mathrm{cbv}} \end{array}$$

The mapping $\cdot^{\mathrm{cbn}}$ corresponds to Levy's (see Def. 2.16), so suffers from the same problem of explosion of syntax as Ex. 2.17. To achieve the simulation results of [7] (see Prop. 2 there), reduction inside thunked terms is explicitly allowed, which is not possible in CBPV.

As to the simulation of CBV reduction in the Bang calculus, terms in function position are forced, and abstractions are thunked; notice that $\cdot^{\mathrm{cbv}}$ maps $\lambda$-values to Bang values. Take a redex $(\lambda x.M)\,N$ and assume that $N$ is not a value, then under CBV reduction, $N$ needs to be evaluated. Under $\cdot^{\mathrm{cbv}}$ we have

$$((\lambda x.M)\,N)^{\mathrm{cbv}} \triangleq \{\lambda x.M^{\mathrm{cbv}}\}!\,N^{\mathrm{cbv}} \to (\lambda x.M^{\mathrm{cbv}})\,N^{\mathrm{cbv}}$$

In the Bang calculus, operands can be evaluated (towards a Bang-value, which under $\cdot^{\mathrm{cbv}}$ is a variable or thunked abstraction), so reduction of $N^{\mathrm{cbv}}$ is possible. Since under $\cdot^{\mathrm{cbv}}$ only abstractions are thunked, to simulate $\lambda$'s CBV reduction that stops at abstractions, reduction in thunked terms is not needed, so can be weak.

So both simulation results for CBN and CBV reduction for the $\lambda$-calculus into the Bang calculus strongly depend on language features that are not shared with CBPV. The simulation results we focus on here are with respect to a calculus, like CBPV, where reduction of operands and inside thunked terms is, other than in the Bang calculus, not allowed, and thereby less easily achieved. So [7]'s results cannot be compared to those we study here. Moreover, types are not used at all in [7].

# 3 The Calculus of Delayed Reductions

In this section we will define the Calculus of Delayed Reductions (CDR), our variant of CBPV, for which it will be possible to define interpretations of the CBN/CBV $\lambda$-calculus that respect single-step reduction, by addressing a number of the issues we mentioned above. The main thing defined differently 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, together with defining a notion of term-substitution $M\lceil V/x\rfloor$ on CDR-terms essentially as normal, with the exception of the case $x!\lceil\{M\}/x\rfloor$ 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 defined using pattern matching, and is comparable to dropping the construct *ret* when contracting an $F$-redex.

**Definition 3.1** (CDR) *i*) Terms of the *Calculus of Delayed Reductions* (CDR) are defined through the grammar:

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

*ii*) The operation of substitution $M\lceil V/x\rfloor$ on CDR-terms is defined as follows:

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

*iii*) The basic reduction rules of CDR are defined using this substitution:

$$(C'): \quad (\lambda x.M)\, V \to_{\mathrm{D}} M\lceil V/x\rfloor$$
$$(F'): \quad x := ret\, V; N \to_{\mathrm{D}} N\lceil V/x\rfloor$$

The CDR-evaluation contexts are (as before) defined through: $\mathsf{C} ::= \lceil\,\rfloor \mid \mathsf{C}V \mid x := \mathsf{C}; M$.
CDR-reduction is defined through:

$$M \to_{\mathrm{D}} N \;\Rightarrow\; \mathsf{C}\lceil M\rfloor \to_{\mathrm{D}} \mathsf{C}\lceil N\rfloor$$

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

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

(as in Def. **??**). We use $\to_{\mathrm{D}}$ for this notion.

Notice the absence of rule $(U)$. Also, observe that only head-variables (that appear in computation position) of (sub)terms are forced, and all other (unforced) occurrences of variables are values. As with $\to_{\mathrm{P}}$, reduction through $\to_{\mathrm{D}}$ is deterministic.

In [16], Levy states "*What our categorical account will not provide is an alternative motivation for CBPV. We do not believe that CBPV can be motivated from a purely categorical perspective, the operational perspective is essential.*" Given the strong relationship between CBPV and CDR, this also applies to the latter.

Type assignment for this variant is inherited from CBPV, so is defined using the rules of Def. 2.4.

**Definition 3.2** Type assignment for CDR is defined through the following inference system, using the CBPV types.

$$(axiom): \frac{}{\Gamma, x{:}A \vdash_{\mathrm{V}} x : A} \qquad (thunk): \frac{\Gamma \vdash_{\mathrm{C}} M : \mathsf{A}}{\Gamma \vdash_{\mathrm{V}} \{M\} : U\mathsf{A}}\ (M \text{ not a forced variable})$$

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

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

Notice that only rules $(thunk)$ and $(force)$ are different from Def. 2.4.

Subject reduction follows easily, after showing first that our notion of substitution preserves assignable types.

*Lemma 3.3* *i) If* $\Gamma, x{:}B \vdash_{\mathrm{V}} W : A$ *and* $\Gamma \vdash_{\mathrm{V}} V : B$, *then* $\Gamma \vdash_{\mathrm{V}} W\lceil V/x\rfloor : A$.
*ii) If* $\Gamma, x{:}B \vdash_{\mathrm{C}} M : \mathsf{A}$ *and* $\Gamma \vdash_{\mathrm{V}} V : B$, *then* $\Gamma \vdash_{\mathrm{C}} M\lceil V/x\rfloor : \mathsf{A}$.

*Proof:* By simultaneous induction on the definition of '$\cdot\lceil V/x\rfloor$'. □

**Theorem 3.4** *If* $\Gamma \vdash_{\mathrm{C}} M : \mathsf{A}$, *and* $M \to_{\mathrm{D}}^{*} N$, *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 $M \equiv (\lambda x.P)\,V$ and $N \equiv P\lceil V/x \rfloor$. 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 $\Gamma \vdash_{\mathrm{C}} P\lceil V/x \rfloor : \mathsf{A}$ by Lem. 3.3.

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

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

so in particular $\Gamma, x{:}B \vdash_{\mathrm{C}} P : \mathsf{A}$ and $\Gamma \vdash_{\mathrm{V}} V : B$. Then $\Gamma \vdash_{\mathrm{C}} P\lceil V/x \rfloor : \mathsf{A}$ by Lem. 3.3.

$P \to_{\mathrm{D}} Q \Rightarrow M \equiv PV \to_{\mathrm{D}} QV \equiv N$: Then there exists $B$ such that the derivation for $\Gamma \vdash_{\mathrm{C}} M : \mathsf{A}$ is constructed as follows:

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

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

$P \to_{\mathrm{D}} Q \Rightarrow M \equiv x := P; R \to_{\mathrm{D}} x := Q; R \equiv N$: Then there exists $B$ such that the derivation for $\Gamma \vdash_{\mathrm{C}} M : \mathsf{A}$ is constructed as follows:

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

By induction we have $\Gamma \vdash_{\mathrm{C}} Q : \mathsf{F}B$; the result follows by rule $(seq)$. □

We can also show that in principle the CK-machine implements reduction in CDR, but need to change its definition first, adapting it to the different notion of term substitution.

**Definition 3.5** (CK$^{\mathrm{D}}$-MACHINE) The reduction relation $\to^{\mathrm{D}}_{\mathrm{CK}}$ on configurations $\langle M \,|\, S \rangle$ with $M \in$ CDR and $S$ a stack of CDR values, is defined as $\to_{\mathrm{CK}}$ in Def. 2.8, but changing the first and fourth case:

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

We define $=^{\mathrm{D}}_{\mathrm{CK}}$ as the equivalence relation generated by $\to^{\mathrm{D}}_{\mathrm{CK}}$.

Notice that the rule $\langle \{M\}! \,|\, S \rangle \to \langle M \,|\, S \rangle$ has been omitted.

The interpretation result now becomes:

**Theorem 3.6** *If $M \to_{\mathrm{D}} N$, then $\langle M \,|\, S \rangle =^{\mathrm{D}}_{\mathrm{CK}} \langle N \,|\, S \rangle$.*

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

$(\lambda x.M)\,V \to_{\mathrm{D}} M\lceil V/x\rfloor$: $\langle (\lambda x.M)\,V \,|\, S\rangle \to^{\mathrm{D}}_{\mathrm{CK}} \langle \lambda x.M \,|\, V : S\rangle \to^{\mathrm{D}}_{\mathrm{CK}} \langle M\lceil V/x\rfloor \,|\, S\rangle$

$x := ret\,V; M \to_{\mathrm{D}} M\{V/x\}$: $\langle x := ret\,V; M \,|\, S\rangle \to^{\mathrm{D}}_{\mathrm{CK}} \langle ret\,V \,|\, x := [\,]; M : S\rangle \to^{\mathrm{D}}_{\mathrm{CK}}$
$\langle M\lceil V/x\rfloor \,|\, S\rangle$

$M \to_{\mathrm{P}} N \Rightarrow MV \to_{\mathrm{D}} NV$: $\langle MV \,|\, S\rangle \to^{\mathrm{D}}_{\mathrm{CK}} \langle M \,|\, V : S\rangle =^{\mathrm{D}}_{\mathrm{CK}} (IH)\ \langle N \,|\, V : S\rangle \leftarrow^{\mathrm{D}}_{\mathrm{CK}} \langle NV \,|\, S\rangle$

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

Notice that, as for Thm. 2.9, we cannot show that reduction is preserved by reduction.

As above, we now have that CDR-reduction is preserved by the interpretation $[\![\cdot]\!]^{\mathrm{D}}_{\mathrm{CK}}$.

*Corollary 3.7* *If* $M \to^*_{\mathrm{D}} N$, *then* $[\![M]\!]^{\mathrm{D}}_{\mathrm{CK}} =^{\mathrm{D}}_{\mathrm{CK}} [\![N]\!]^{\mathrm{D}}_{\mathrm{CK}}$.

### 3.1 A CBV-interpretation of $\Lambda$ in CDR

We will now define our CBV interpretation of pure $\lambda$-terms into CDR and show that it *does* respect single-step CBV-reduction, and not just equality.

**Definition 3.8** The CBV interpretation $\lceil\!\cdot\!\rfloor^{\lambda}_{\mathrm{v}}$ of $\lambda$-terms into CDR is defined through:

$$\lceil\!x\!\rfloor^{\lambda}_{\mathrm{v}} \triangleq ret\,x \qquad \lceil\!\lambda x.M\!\rfloor^{\lambda}_{\mathrm{v}} \triangleq ret\{\lambda x.\lceil\!M\!\rfloor^{\lambda}_{\mathrm{v}}\}$$

$$\lceil\!zN\!\rfloor^{\lambda}_{\mathrm{v}} \triangleq y := \lceil\!N\!\rfloor^{\lambda}_{\mathrm{v}}; z!y$$
$$\lceil\!(\lambda z.M)\,N\!\rfloor^{\lambda}_{\mathrm{v}} \triangleq y := \lceil\!N\!\rfloor^{\lambda}_{\mathrm{v}}; (\lambda z.\lceil\!M\!\rfloor^{\lambda}_{\mathrm{v}})y$$
$$\lceil\!MN\!\rfloor^{\lambda}_{\mathrm{v}} \triangleq x := \lceil\!M\!\rfloor^{\lambda}_{\mathrm{v}}; y := \lceil\!N\!\rfloor^{\lambda}_{\mathrm{v}}; x!y \quad (M \text{ not a value})$$

Our CBV interpretation deals with terms basically the same way as Levy's (see Def. 2.11), but for the fact that we place the image of the $\lambda$-value $V$ in the interpretation of $VN$ *directly*, so without using assignment; this is done using pattern matching, so without using (implicit) substitution, since $V$ is either a variable or an abstraction.

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

*Lemma 3.9* (SUBSTITUTION LEMMA FOR $\lceil\!\cdot\!\rfloor^{\lambda}_{\mathrm{v}}$) *i)* $\lceil\!M\!\rfloor^{\lambda}_{\mathrm{v}}\lceil w/z\rfloor = \lceil\!M\{w/z\}\!\rfloor^{\lambda}_{\mathrm{v}}$.
*ii)* $\lceil\!M\!\rfloor^{\lambda}_{\mathrm{v}}\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor = \lceil\!M\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}}$.

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

*ii*) $M \equiv z$: $\lceil\!z\!\rfloor^{\lambda}_{\mathrm{v}}\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor \triangleq (ret\,z)\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor = ret\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\} \triangleq \lceil\!\lambda w.R\!\rfloor^{\lambda}_{\mathrm{v}} = \lceil\!z\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}}$

$M \equiv u$ *and* $u \neq z$: $\lceil\!u\!\rfloor^{\lambda}_{\mathrm{v}}\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor \triangleq (ret\,u)\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor = ret\,u \triangleq \lceil\!u\!\rfloor^{\lambda}_{\mathrm{v}} = \lceil\!u\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}}$

$M \equiv \lambda u.N$: $\lceil\!\lambda u.N\!\rfloor^{\lambda}_{\mathrm{v}}\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor \triangleq ret\{\lambda u.(\lceil\!N\!\rfloor^{\lambda}_{\mathrm{v}}\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor) = (IH)$
$ret\{\lambda u.\lceil\!N\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}}\} = \lceil\!\lambda u.N\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}}$

$M \equiv xQ$: $\lceil\!xQ\!\rfloor^{\lambda}_{\mathrm{v}}\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor \triangleq y := \lceil\!Q\!\rfloor^{\lambda}_{\mathrm{v}}\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor; x!y = (IH)$
$y := \lceil\!Q\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}}; x!y \triangleq \lceil\!xQ\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}} = \lceil\!(xQ)\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}}$

$M \equiv (\lambda x.P)\,Q$: $\lceil\!(\lambda x.P)\,Q\!\rfloor^{\lambda}_{\mathrm{v}}\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor \triangleq (y := \lceil\!Q\!\rfloor^{\lambda}_{\mathrm{v}}; (\lambda x.\lceil\!P\!\rfloor^{\lambda}_{\mathrm{v}})y)\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor \triangleq$
$y := \lceil\!Q\!\rfloor^{\lambda}_{\mathrm{v}}\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor; (\lambda x.\lceil\!P\!\rfloor^{\lambda}_{\mathrm{v}}\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor)\,y = (IH)$
$y := \lceil\!Q\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}}; (\lambda x.\lceil\!P\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}})\,y \triangleq \lceil\!(\lambda x.P)\{\lambda w.R/z\}\,Q\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}} = \lceil\!((\lambda x.P)\,Q)\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}}$

$M \equiv PQ$, $P$ *not a* $\lambda$*-value*: $\lceil\!PQ\!\rfloor^{\lambda}_{\mathrm{v}}\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor \triangleq$
$x := ; y := \lceil\!P\!\rfloor^{\lambda}_{\mathrm{v}}; x!y\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor\lceil\!Q\!\rfloor^{\lambda}_{\mathrm{v}}\lceil\{\lambda w.\lceil\!R\!\rfloor^{\lambda}_{\mathrm{v}}\}/z\rfloor = (IH)$
$x := \lceil\!P\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}}; y := \lceil\!Q\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}}; x!y \triangleq \lceil\!P\{\lambda w.R/z\}Q\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}} = \lceil\!(PQ)\{\lambda w.R/z\}\!\rfloor^{\lambda}_{\mathrm{v}}$ □

Using this lemma, we can now show:

**Theorem 3.10** *If $M \to_{\mathrm{v}} N$, then $\llbracket M \rrbracket^{\lambda}_{\mathrm{v}} \to^{*}_{\mathrm{D}} \llbracket N \rrbracket^{\lambda}_{\mathrm{v}}$.*

*Proof:* $(\lambda x.M)\, z \to_{\mathrm{v}} M\{z/x\}$: $\llbracket (\lambda x.M)\, z \rrbracket^{\lambda}_{\mathrm{v}} \triangleq y := \mathit{ret}\, z; (\lambda x.\llbracket M \rrbracket^{\lambda}_{\mathrm{v}})\, y \to_{\mathrm{D}} (\lambda x.\llbracket M \rrbracket^{\lambda}_{\mathrm{v}})\, z \to_{\mathrm{D}} \llbracket M \rrbracket^{\lambda}_{\mathrm{v}} \lceil z/x \rfloor = (3.9)\ \llbracket M\{z/x\} \rrbracket^{\lambda}_{\mathrm{v}}$

$(\lambda x.M)\, (\lambda z.R) \to_{\mathrm{v}} M\{\lambda z.R/x\}$: $\llbracket (\lambda x.M)\, (\lambda z.R) \rrbracket^{\lambda}_{\mathrm{v}} \triangleq y := \llbracket \lambda z.R \rrbracket^{\lambda}_{\mathrm{v}}; (\lambda x.\llbracket M \rrbracket^{\lambda}_{\mathrm{v}}) y \triangleq y := \mathit{ret}\{\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}; (\lambda x.\llbracket M \rrbracket^{\lambda}_{\mathrm{v}}) y \to_{\mathrm{D}} (\lambda x.\llbracket M \rrbracket^{\lambda}_{\mathrm{v}})\{\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\} \to_{\mathrm{D}} \llbracket M \rrbracket^{\lambda}_{\mathrm{v}} \lceil \{\lambda z.\llbracket R \rrbracket^{\lambda}_{\mathrm{v}}\}/x \rfloor = (3.9)\ \llbracket M\{\lambda z.R/x\} \rrbracket^{\lambda}_{\mathrm{v}}$

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

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

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

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

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

Notice that this result was shown for reduction in CDR, in contrast to Thm. 2.13 which was shown for equality in CBPV.

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

**Theorem 3.11** *If $\Gamma \vdash_{\lambda} M : A$, then $\overline{\Gamma} \vdash_{\mathrm{C}} \llbracket M \rrbracket^{\lambda}_{\mathrm{v}} : \mathsf{F}\overline{A}$.*

*Proof:* By induction on definition of $\vdash_{\lambda}$.

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

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

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

$(\to I)$: Then $A = B \to C$, $M \equiv \lambda x.N$, and $\Gamma, x{:}B \vdash_{\lambda} N : C$; 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:

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

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 $\Gamma \vdash_{\lambda} P : B \to A$ and $\Gamma \vdash_{\lambda} Q : B$. 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:

$$\dfrac{\boxed{\phantom{x}}\ \overline{\Gamma}\vdash_{\mathrm{C}} [\![Q]\!]^{\lambda}_{\mathrm{v}} : \mathsf{F}\overline{B} \qquad \dfrac{\dfrac{}{\overline{\Gamma},y{:}\overline{B}\vdash_{\mathrm{C}} z! : \overline{B}\to\mathsf{F}\overline{A}}\,(force) \quad \dfrac{}{\overline{\Gamma},y{:}\overline{B}\vdash_{\mathrm{v}} y : \overline{B}}\,(axiom)}{\overline{\Gamma},y{:}\overline{B}\vdash_{\mathrm{C}} z!y : \mathsf{F}\overline{A}}\,(appl)}{\overline{\Gamma}\vdash_{\mathrm{C}} y := [\![Q]\!]^{\lambda}_{\mathrm{v}}; z!y : \mathsf{F}\overline{A}}\,(seq)$$

Notice that $[\![zQ]\!]^{\lambda}_{\mathrm{v}} \stackrel{\Delta}{=} y := [\![Q]\!]^{\lambda}_{\mathrm{v}}; z!y$.

$P \equiv \lambda z.R$: From $\overline{\Gamma}\vdash_{\mathrm{C}} [\![P]\!]^{\lambda}_{\mathrm{v}} : \mathsf{F}\overline{B\to A} \stackrel{\Delta}{=} \overline{\Gamma}\vdash_{\mathrm{C}} ret\{\lambda z.[\![R]\!]^{\lambda}_{\mathrm{v}}\} : \mathsf{F}U(\overline{B}\to\mathsf{F}\overline{A})$, which is shaped like:

$$\dfrac{\dfrac{\dfrac{\boxed{\phantom{x}}\ \overline{\Gamma}\vdash_{\mathrm{C}} \lambda z.[\![R]\!]^{\lambda}_{\mathrm{v}} : \overline{B}\to\mathsf{F}\overline{A}}{\overline{\Gamma}\vdash_{\mathrm{v}} \{\lambda z.[\![R]\!]^{\lambda}_{\mathrm{v}}\} : U(\overline{B}\to\mathsf{F}\overline{A})}\,(thunk)}{\overline{\Gamma}\vdash_{\mathrm{C}} ret\{\lambda z.[\![R]\!]^{\lambda}_{\mathrm{v}}\} : \mathsf{F}U(\overline{B}\to\mathsf{F}\overline{A})}\,(ret)$$

we know that in a subderivation $\overline{\Gamma}\vdash_{\mathrm{C}} \lambda z.[\![R]\!]^{\lambda}_{\mathrm{v}} : \overline{B}\to\mathsf{F}\overline{A}$ is shown, with which we can construct:

$$\dfrac{\boxed{\phantom{x}}\ \overline{\Gamma}\vdash_{\mathrm{C}} [\![Q]\!]^{\lambda}_{\mathrm{v}} : \mathsf{F}\overline{B} \qquad \dfrac{\dfrac{\boxed{\phantom{x}}\ \overline{\Gamma}\vdash_{\mathrm{C}} \lambda z.[\![R]\!]^{\lambda}_{\mathrm{v}} : \overline{B}\to\mathsf{F}\overline{A}}{\overline{\Gamma},y{:}\overline{B}\vdash_{\mathrm{C}} \lambda z.[\![R]\!]^{\lambda}_{\mathrm{v}} : \overline{B}\to\mathsf{F}\overline{A}}\,(Wk) \quad \dfrac{}{\overline{\Gamma},y{:}\overline{B}\vdash_{\mathrm{v}} y : \overline{B}}\,(axiom)}{\overline{\Gamma},y{:}\overline{B}\vdash_{\mathrm{C}} (\lambda z.[\![R]\!]^{\lambda}_{\mathrm{v}})\,y : \mathsf{F}\overline{A}}\,(appl)}{\overline{\Gamma}\vdash_{\mathrm{C}} y := [\![Q]\!]^{\lambda}_{\mathrm{v}}; (\lambda z.[\![R]\!]^{\lambda}_{\mathrm{v}})y : \mathsf{F}\overline{A}}\,(seq)$$

Notice that $[\![(\lambda z.R)\,Q]\!]^{\lambda}_{\mathrm{v}} \stackrel{\Delta}{=} y := [\![Q]\!]^{\lambda}_{\mathrm{v}}; (\lambda z.[\![R]\!]^{\lambda}_{\mathrm{v}})y$.

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

$$\dfrac{\dfrac{\boxed{\phantom{x}}\ \overline{\Gamma}\vdash_{\mathrm{C}} [\![Q]\!]^{\lambda}_{\mathrm{v}} : \mathsf{F}\overline{B}}{\overline{\Gamma},x{:}U(\overline{B}\to\mathsf{F}\overline{A})\vdash_{\mathrm{C}} [\![Q]\!]^{\lambda}_{\mathrm{v}} : \mathsf{F}\overline{B}}\,(Wk) \quad \vdots \qquad \dfrac{\dfrac{\dfrac{}{\Gamma'\vdash_{\mathrm{v}} x : U(\overline{B}\to\mathsf{F}\overline{A})}\,(axiom)}{\Gamma'\vdash_{\mathrm{C}} x! : \overline{B}\to\mathsf{F}\overline{A}}\,(force) \quad \dfrac{}{\Gamma'\vdash_{\mathrm{v}} y : \overline{B}}\,(axiom)}{\overline{\Gamma},x{:}U(\overline{B}\to\mathsf{F}\overline{A}),y{:}\overline{B}\vdash_{\mathrm{C}} x!y : \mathsf{F}\overline{A}}\,(appl)}{\overline{\Gamma},x{:}U(\overline{B}\to\mathsf{F}\overline{A})\vdash_{\mathrm{C}} y := [\![Q]\!]^{\lambda}_{\mathrm{v}}; x!y : \mathsf{F}\overline{A}}\,(seq)}{\overline{\Gamma}\vdash_{\mathrm{C}} x := [\![P]\!]^{\lambda}_{\mathrm{v}}; y := [\![Q]\!]^{\lambda}_{\mathrm{v}}; x!y : \mathsf{F}\overline{A}}\,(seq)$$

Notice that $[\![PQ]\!]^{\lambda}_{\mathrm{v}} \stackrel{\Delta}{=} x := [\![P]\!]^{\lambda}_{\mathrm{v}}; y := [\![Q]\!]^{\lambda}_{\mathrm{v}}; x!y$. □

### 3.2 A CBN-interpretation of $\Lambda$ in CDR

We can achieve a stronger result for CBN-reduction as well, as we will now show; this is where the new substitution comes into play. First we modify the interpretation of Def. 2.16 to fit CDR.

**Definition 3.12** The CBN interpretation $[\![\cdot]\!]^{\lambda}_{\mathrm{N}}$ of $\lambda$-terms into CDR is defined through:

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

Notice that this corresponds to Levy's CBN-interpretation in Def. 2.16, but for the fact that only head-variables (that appear in computation position) are forced; in the application case we avoid to define $[\![Mx]\!]^{\lambda}_{\mathrm{N}} = [\![M]\!]^{\lambda}_{\mathrm{N}}\{x!\}$, but directly write $[\![M]\!]^{\lambda}_{\mathrm{N}} x$, and that we do not add unnecessary occurrences of $\{\cdot\}!$.

Now, other than in Example 2.17 we get:

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

This interpretation respects the term substitutions:

*Lemma 3.13* (Substitution lemma for $\llbracket\cdot\rrbracket^{\lambda}_{\mathrm{N}}$) *i)* $\llbracket M\rrbracket^{\lambda}_{\mathrm{N}}\lceil w/z\rfloor = \llbracket M\{w/z\}\rrbracket^{\lambda}_{\mathrm{N}}$.
*ii)* $\llbracket M\rrbracket^{\lambda}_{\mathrm{N}}\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/x\rfloor = \llbracket M\{N/x\}\rrbracket^{\lambda}_{\mathrm{N}}$, *if $N$ is not a variable.*

*Proof: i)* By straightforward induction on the definition of $\cdot\lceil\cdot/\cdot\rfloor$.
*ii)* $M\equiv z$: $\llbracket z\rrbracket^{\lambda}_{\mathrm{N}}\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor = z!\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor = \llbracket N\rrbracket^{\lambda}_{\mathrm{N}} = \llbracket z\{N/z\}\rrbracket^{\lambda}_{\mathrm{N}}$
$M\equiv u$ *and* $u\neq z$: $\llbracket u\rrbracket^{\lambda}_{\mathrm{N}}\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor = (u!)\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor = u! = \llbracket u\rrbracket^{\lambda}_{\mathrm{N}} = \llbracket u\{N/z\}\rrbracket^{\lambda}_{\mathrm{N}}$
$M\equiv \lambda u.R$: $\llbracket\lambda u.R\rrbracket^{\lambda}_{\mathrm{N}}\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor = (\lambda u.\llbracket R\rrbracket^{\lambda}_{\mathrm{N}})\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor = \lambda u.(\llbracket R\rrbracket^{\lambda}_{\mathrm{N}}\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor)$
$= (IH)\quad \lambda u.\llbracket R\{N/z\}\rrbracket^{\lambda}_{\mathrm{N}} = \llbracket\lambda u.R\{N/z\}\rrbracket^{\lambda}_{\mathrm{N}}$
$M\equiv Pz$: $\llbracket Pz\rrbracket^{\lambda}_{\mathrm{N}}\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor = (\llbracket P\rrbracket^{\lambda}_{\mathrm{N}}z)\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor = [\![P]\!]\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\} = (IH)$
$\llbracket P\{N/z\}\rrbracket^{\lambda}_{\mathrm{N}}\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\} = \llbracket P\{N/z\}N\rrbracket^{\lambda}_{\mathrm{N}} = \llbracket(Pz)\{N/z\}\rrbracket^{\lambda}_{\mathrm{N}}$
$M\equiv Px$, $x\neq z$: $\llbracket Px\rrbracket^{\lambda}_{\mathrm{N}}\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor = (\llbracket P\rrbracket^{\lambda}_{\mathrm{N}}x)\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor = [\![P]\!]\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor\,x = (IH)$
$\llbracket P\{N/z\}\rrbracket^{\lambda}_{\mathrm{N}}x = \llbracket P\{N/z\}x\rrbracket^{\lambda}_{\mathrm{N}} = \llbracket(Px)\{N/z\}\rrbracket^{\lambda}_{\mathrm{N}}$
$M\equiv PQ$: $\llbracket PQ\rrbracket^{\lambda}_{\mathrm{N}}\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor = (\llbracket P\rrbracket^{\lambda}_{\mathrm{N}}\{\llbracket Q\rrbracket^{\lambda}_{\mathrm{N}}\})\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor =$
$\llbracket P\rrbracket^{\lambda}_{\mathrm{N}}\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor\{\llbracket Q\rrbracket^{\lambda}_{\mathrm{N}}\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/z\rfloor\} = (IH)\ \llbracket P\{N/z\}\rrbracket^{\lambda}_{\mathrm{N}}\{\llbracket Q\{N/z\}\rrbracket^{\lambda}_{\mathrm{N}}\} =$
$\llbracket P\{N/z\}Q\{N/z\}\rrbracket^{\lambda}_{\mathrm{N}} = \llbracket(PQ)\{N/z\}\rrbracket^{\lambda}_{\mathrm{N}}$ □

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

**Theorem 3.14** *If $M\to_{\mathrm{N}} N$, then $\llbracket M\rrbracket^{\lambda}_{\mathrm{N}}\to^{*}_{\mathrm{D}}\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}$.*

*Proof:* $(\lambda x.M)\,y\to_{\mathrm{N}} M\{y/x\}$: $\llbracket(\lambda x.M)\,y\rrbracket^{\lambda}_{\mathrm{N}}\triangleq(\lambda x.\llbracket M\rrbracket^{\lambda}_{\mathrm{N}})\,y\to_{\mathrm{D}}\llbracket M\rrbracket^{\lambda}_{\mathrm{N}}\lceil y/x\rfloor = (3.13)$
$\llbracket M\{y/x\}\rrbracket^{\lambda}_{\mathrm{N}}$
$(\lambda x.M)\,N\to_{\mathrm{N}} M\{N/x\}$, *$N$ not a variable*: $\llbracket(\lambda x.M)\,N\rrbracket^{\lambda}_{\mathrm{N}}\triangleq(\lambda x.\llbracket M\rrbracket^{\lambda}_{\mathrm{N}})\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}\to_{\mathrm{D}}$
$\llbracket M\rrbracket^{\lambda}_{\mathrm{N}}\lceil\{\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\}/x\rfloor = (3.13)\ \llbracket M\{N/x\}\rrbracket^{\lambda}_{\mathrm{N}}$
$M\to_{\mathrm{N}} N\Rightarrow Mx\to_{\mathrm{N}} Nx$: $\llbracket Mx\rrbracket^{\lambda}_{\mathrm{N}}\triangleq\llbracket M\rrbracket^{\lambda}_{\mathrm{N}}x\to^{*}_{\mathrm{D}}(IH)\ \llbracket N\rrbracket^{\lambda}_{\mathrm{N}}x\triangleq\llbracket Nx\rrbracket^{\lambda}_{\mathrm{N}}$
$M\to_{\mathrm{N}} N\Rightarrow MP\to_{\mathrm{N}} NP$, *with $P$ not a variable*: $\llbracket MP\rrbracket^{\lambda}_{\mathrm{N}}\triangleq\llbracket M\rrbracket^{\lambda}_{\mathrm{N}}\{\llbracket P\rrbracket^{\lambda}_{\mathrm{N}}\}\to^{*}_{\mathrm{D}}(IH)$
$\llbracket N\rrbracket^{\lambda}_{\mathrm{N}}\{\llbracket P\rrbracket^{\lambda}_{\mathrm{N}}\}\triangleq\llbracket NP\rrbracket^{\lambda}_{\mathrm{N}}$ □

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 3.15** (Simple type interpretation [16]) 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 3.16** *If $\Gamma\vdash_{\lambda} M:A$, 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:

$$\frac{}{\underline{\Gamma} \vdash_{\mathrm{C}} x! : \underline{A}}\ (\mathit{force})$$

and $[\![x]\!]^{\lambda}_{\mathrm{N}} \triangleq x!$.

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

$$\frac{\boxed{\quad}}{\underline{\Gamma} \vdash_{\mathrm{C}} \lambda x.[\![N]\!]^{\lambda}_{\mathrm{N}} : U\underline{B} \to \underline{C}}\ (\mathit{abstr})$$

and $[\![\lambda x.N]\!]^{\lambda}_{\mathrm{N}} \triangleq \lambda x.[\![N]\!]^{\lambda}_{\mathrm{N}}$ and $\underline{B \to C} = U\underline{B} \to \underline{C}$.

$(\to E)$: Then $M \equiv PQ$, and there exists $B$ such that $\Gamma \vdash_{\lambda} P : B \to A$ and $\Gamma \vdash_{\lambda} Q : B$. By induction, we get $\underline{\Gamma} \vdash_{\mathrm{C}} [\![P]\!]^{\lambda}_{\mathrm{N}} : \underline{B \to A}$; also, $\underline{B \to A} = U\underline{B} \to \underline{A}$. We have two cases:

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

$$\frac{\boxed{\quad} \qquad \frac{}{\underline{\Gamma} \vdash_{\mathrm{V}} x : U\underline{B}}\ (Ax)}{\underline{\Gamma} \vdash_{\mathrm{C}} [\![P]\!]^{\lambda}_{\mathrm{N}} x : \underline{A}}\ (\mathit{appl})$$

and $[\![Px]\!]^{\lambda}_{\mathrm{N}} \triangleq [\![P]\!]^{\lambda}_{\mathrm{N}} x$.

$Q \neq x$: By induction, we have $\underline{\Gamma} \vdash_{\mathrm{C}} [\![Q]\!]^{\lambda}_{\mathrm{N}} : \underline{B}$, and can construct:

$$\frac{\boxed{\quad} \qquad \frac{\boxed{\quad}}{\underline{\Gamma} \vdash_{\mathrm{V}} \{[\![Q]\!]^{\lambda}_{\mathrm{N}}\} : U\underline{B}}\ (\mathit{thunk})}{\underline{\Gamma} \vdash_{\mathrm{C}} [\![P]\!]^{\lambda}_{\mathrm{N}}\{[\![Q]\!]^{\lambda}_{\mathrm{N}}\} : \underline{A}}\ (\mathit{appl})$$

and $[\![PQ]\!]^{\lambda}_{\mathrm{N}} \triangleq [\![P]\!]^{\lambda}_{\mathrm{N}}\{[\![Q]\!]^{\lambda}_{\mathrm{N}}\}$. □

# 4 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 [4]. 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 $\alpha, \beta, \gamma$, *etc.*, label the types of the conclusions. The syntax of $\overline{\lambda}\mu\tilde{\mu}$ has three different categories: commands, terms, and environments. Commands $c$ form the computational units in $\overline{\lambda}\mu\tilde{\mu}$ and are composed of a pair $\langle t \,|\, e \rangle$ of a term $t$ and its environment $e$ that can interact.

**Definition 4.1** (Commands, Terms, and Environments [4]) Using an infinite countable set of *term variables* $\{x, y, z, \dots\}$ and an infinite countable set of *environment variables* $\{\alpha, \beta, \gamma, \dots\}$, the three categories of expressions in $\overline{\lambda}\mu\tilde{\mu}$ are defined by:

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

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

With conventional notations about environments (*i.e.* seeing environments as terms with a hole), $t \cdot e$ can be thought of as $e[[\ ]\ t]$, and the environment $t_1 \cdot (\cdots (t_n \cdot \alpha) \cdots)$ (we can omit

these brackets and write $t_1 \cdots t_n \cdot \alpha$) as a *stack* (see Example 4.8); $\mu\alpha.c$ is inherited from $\lambda\mu$, as is $\langle t \,|\, \alpha \rangle$ which corresponds to $\lambda\mu$'s *naming* construct $[\alpha]t$, giving name $\alpha$ to the implicit output name of $t$; the construct $\tilde{\mu}x.c$ can be thought of as $\mathsf{let}\ x = [\,]\ \mathsf{in}\ c$.

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

**Definition 4.2** (Reduction in $\overline{\lambda}\mu\tilde{\mu}$ [4, 12]) Let $c\{e/\beta\}$ stand for the implicit substitution of the free occurrences of the environment variable $\beta$ by the environment $e$, and $c\{t/x\}$ for that of $x$ by the term $t$. The reduction rules are defined by:

*logical rules*

$$(\lambda): \quad \langle \lambda x.t_1 \,|\, t_2 \cdot e \rangle \to \langle t_2 \,|\, \tilde{\mu}x.\langle t_1 \,|\, e \rangle \rangle$$
$$(\mu): \quad \langle \mu\beta.c \,|\, e \rangle \to c\{e/\beta\}$$
$$(\tilde{\mu}): \quad \langle t \,|\, \tilde{\mu}x.c \rangle \to c\{t/x\}$$

*extensional rules*

$$(\eta): \lambda x.\mu\beta.\langle t \,|\, x \cdot \beta \rangle \to t \quad (x, \beta \notin fv(t))$$
$$(\eta\mu): \quad \mu\alpha.\langle t \,|\, \alpha \rangle \to t \quad (\alpha \notin fv(t))$$
$$(\eta\tilde{\mu}): \quad \tilde{\mu}x.\langle x \,|\, e \rangle \to e \quad (x \notin fv(e))$$

*contextual rules*

$$t \to t' \Rightarrow \begin{cases} \langle t \,|\, e \rangle \to \langle t' \,|\, e \rangle \\ \lambda x.t \to \lambda x.t' \\ t \cdot e \to t' \cdot e \end{cases}$$
$$e \to e' \Rightarrow \begin{cases} \langle t \,|\, e \rangle \to \langle t \,|\, e' \rangle \\ t \cdot e \to t \cdot e' \end{cases}$$
$$c \to c' \Rightarrow \begin{cases} \mu\beta.c \to \mu\beta.c' \\ \tilde{\mu}x.c \to \tilde{\mu}x.c' \end{cases}$$

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

Notice that rules $(\lambda)$, $(\mu)$, and $(\tilde{\mu})$ reduce commands to commands, rules $(\eta)$ and $(\eta\mu)$ reduce a term to a term, and rule $(\eta\tilde{\mu})$ reduces a environment to a environment. Apart from Theorem 4.7, the extensional rules play no role in this paper. Not all commands can be reduced: *e.g.* $\langle x \,|\, \alpha \rangle$, $\langle \lambda x.t \,|\, \alpha \rangle$ and $\langle x \,|\, t \cdot e \rangle$ 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 *environment construction* $t \cdot e$, where a term with a hole is built, offering the operand $t$ and the continuation $e$. The main operators are $\mu$ and $\tilde{\mu}$ abstraction, which, in a sense, respectively, correspond to (delayed) substitution (parameter call) and to environment call.

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

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

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

**Definition 4.3** (Typing for $\overline{\lambda}\mu\tilde{\mu}$ [4]) Using Curry types (Definition 1.3), type assignment is defined via the rules:

$$(cut): \frac{\Gamma \vdash t : A \mid \Delta \quad \Gamma \mid e : A \vdash \Delta}{\langle t \,|\, e \rangle : \Gamma \vdash \Delta} \qquad (\mu): \frac{c : \Gamma \vdash \alpha{:}A, \Delta}{\Gamma \vdash \mu\alpha.c : A \mid \Delta} \qquad (\tilde{\mu}): \frac{c : \Gamma, x{:}A \vdash \Delta}{\Gamma \mid \tilde{\mu}x.c : A \vdash \Delta}$$

$$(AxR): \frac{}{\Gamma, x{:}A \vdash x : A \mid \Delta} \qquad (AxL): \frac{}{\Gamma \mid \alpha : A \vdash \alpha{:}A, \Delta}$$

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

We write $c : \Gamma \vdash \Delta$, $\Gamma \vdash t : A \mid \Delta$, and $\Gamma \mid e : A \vdash \Delta$ 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)

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

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

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

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 [21]).

In [4] 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 $\mu$-redex, whereas the CBN sub-reduction forbids a $\mu$-reduction when the redex is also a $\tilde{\mu}$-redex; there is no other restriction defined in [4, 12] 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 4.4** (CBV AND CBN REDUCTION FOR $\overline{\lambda}\mu\tilde{\mu}$) *i*) *Values* $V$ are defined by $V ::= x \mid \lambda x.t$, and *stacks*[17] $E$ are defined by $E ::= \alpha \mid t \cdot e$.

*ii*) CBN-reduction $\to_{\mathrm{P}}^{\mathrm{N}}$ is defined by limiting rule $(\mu)$ and restricting the contextual rules:

$$\begin{array}{rll}
(\lambda): & \langle \lambda x.t_1 \mid t_2 \cdot e \rangle \to \langle t_2 \mid \tilde{\mu}x.\langle t_1 \mid e \rangle \rangle & \\
(\mu_n): & \langle \mu\beta.c \mid E \rangle \to c\{E/\beta\} & t \to t' \;\Rightarrow\; \langle t \mid e \rangle \to \langle t' \mid e \rangle \\
(\tilde{\mu}): & \langle t \mid \tilde{\mu}x.c \rangle \to c\{t/x\} & c \to c' \;\Rightarrow\; \mu\beta.c \to \mu\beta.c' \\
(\eta\mu): & \mu\alpha.\langle t \mid \alpha \rangle \to t \quad (\alpha \notin fv(t)) &
\end{array}$$

*iii*) CBV-reduction $\to_{\mathrm{P}}^{\mathrm{V}}$ is defined by limiting rule $(\tilde{\mu})$ and restricting the contextual rules:

[17] In [12], stacks are called *linear evaluation contexts*.

$$
\begin{array}{rll}
(\lambda): & \langle \lambda x.t_1 \,|\, t_2 \cdot e\rangle \to \langle t_2 \,|\, \tilde{\mu}x.\langle t_1 \,|\, e\rangle\rangle & \\
(\mu): & \langle \mu\beta.c \,|\, e\rangle \to c\{e/\beta\} & t \to t' \;\Rightarrow\; \langle t \,|\, e\rangle \to \langle t' \,|\, e\rangle \\
(\tilde{\mu}_v): & \langle V \,|\, \tilde{\mu}x.c\rangle \to c\{V/x\} & c \to c' \;\Rightarrow\; \mu\beta.c \to \mu\beta.c' \\
(\eta\mu): & \mu\alpha.\langle t \,|\, \alpha\rangle \to t \quad (\alpha \notin fv(t)) &
\end{array}
$$

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

Notice that both notions do not permit reduction in environments, are defined by eliminating the same contextual reduction rules, and only differ in rules $(\mu)$ and $(\tilde{\mu})$.

Essentially following [4], an interpretation $[\![\cdot]\!]^{\lambda}_{\mathrm{H}}$ of the $\lambda$-calculus into $\overline{\lambda}\mu\tilde{\mu}$ can be defined as follows:

**Definition 4.5** Interpretation of the $\lambda$-calculus into $\overline{\lambda}\mu\tilde{\mu}$:

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

Notice that $\lambda$-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 4.6* ([1]) $[\![M\{N/x\}]\!]^{\lambda}_{\mathrm{H}} = [\![M]\!]^{\lambda}_{\mathrm{H}}\{[\![N]\!]^{\lambda}_{\mathrm{H}}/x\}$.

Using this result, we can now show that the interpretation respects $\beta$-reduction.

**Theorem 4.7** *i) If $M \to^*_{\beta} N$, then $[\![M]\!]^{\lambda}_{\mathrm{H}} \to^*_{\overline{\lambda}} [\![N]\!]^{\lambda}_{\mathrm{H}}$.*
*ii) If $M \to^*_{\mathrm{N}} N$, then $[\![M]\!]^{\lambda}_{\mathrm{H}} \to^{\mathrm{N}*}_{\overline{\lambda}} [\![N]\!]^{\lambda}_{\mathrm{H}}$.*
*iii) If $M \to^*_{\mathrm{v}} N$, then $[\![M]\!]^{\lambda}_{\mathrm{H}} \to^{\mathrm{v}*}_{\overline{\lambda}} [\![N]\!]^{\lambda}_{\mathrm{H}}$.*

*Proof: i)* $(\lambda x.M)N \to_{\beta} M\{N/x\}$: $[\![(\lambda x.M)N]\!]^{\lambda}_{\mathrm{H}} \triangleq \mu\alpha.\langle [\![\lambda x.M]\!]^{\lambda}_{\mathrm{H}} \,|\, [\![N]\!]^{\lambda}_{\mathrm{H}} \cdot \alpha\rangle \triangleq$
$\mu\alpha.\langle \lambda x.[\![M]\!]^{\lambda}_{\mathrm{H}} \,|\, [\![N]\!]^{\lambda}_{\mathrm{H}} \cdot \alpha\rangle \to_{\overline{\lambda}} (\lambda)\ \mu\alpha.\langle [\![N]\!]^{\lambda}_{\mathrm{H}} \,|\, \tilde{\mu}x.\langle [\![M]\!]^{\lambda}_{\mathrm{H}} \,|\, \alpha\rangle\rangle \to_{\overline{\lambda}} (\tilde{\mu})$
$\mu\alpha.\langle [\![M]\!]^{\lambda}_{\mathrm{H}}\{[\![N]\!]^{\lambda}_{\mathrm{H}}/x\} \,|\, \alpha\rangle \to_{\overline{\lambda}} (\eta\mu)\ [\![M]\!]^{\lambda}_{\mathrm{H}}\{[\![N]\!]^{\lambda}_{\mathrm{H}}/x\}$.

$M \to_{\beta} N \Rightarrow \lambda x.M \to_{\beta} \lambda x.N$: $[\![\lambda x.M]\!]^{\lambda}_{\mathrm{H}} \triangleq \lambda x.[\![M]\!]^{\lambda}_{\mathrm{H}} \to^*_{\overline{\lambda}} (IH)\ \lambda x.[\![N]\!]^{\lambda}_{\mathrm{H}} \triangleq [\![\lambda x.N]\!]^{\lambda}_{\mathrm{H}}$.

$M \to_{\beta} N \Rightarrow MP \to_{\beta} NP$: $[\![MP]\!]^{\lambda}_{\mathrm{H}} \triangleq \mu\alpha.\langle [\![M]\!]^{\lambda}_{\mathrm{H}} \,|\, [\![P]\!]^{\lambda}_{\mathrm{H}} \cdot \alpha\rangle \to^*_{\overline{\lambda}} (IH)\ \mu\alpha.\langle [\![N]\!]^{\lambda}_{\mathrm{H}} \,|\, [\![P]\!]^{\lambda}_{\mathrm{H}} \cdot \alpha\rangle \triangleq [\![NP]\!]^{\lambda}_{\mathrm{H}}$.

$M \to_{\beta} N \Rightarrow PM \to_{\beta} PN$: $[\![PM]\!]^{\lambda}_{\mathrm{H}} \triangleq \mu\alpha.\langle [\![P]\!]^{\lambda}_{\mathrm{H}} \,|\, [\![M]\!]^{\lambda}_{\mathrm{H}} \cdot \alpha\rangle \to^*_{\overline{\lambda}} (IH)\ \mu\alpha.\langle [\![P]\!]^{\lambda}_{\mathrm{H}} \,|\, [\![N]\!]^{\lambda}_{\mathrm{H}} \cdot \alpha\rangle \triangleq [\![PN]\!]^{\lambda}_{\mathrm{H}}$.

*ii)* $(\lambda x.M)N \to_{\mathrm{N}} M\{N/x\}$: As in part (*i*) above; notice that all reduction steps are permitted in $\to_{\mathrm{N}}$.

$M \to_{\mathrm{N}} N \Rightarrow MP \to_{\mathrm{N}} NP$: $[\![MP]\!]^{\lambda}_{\mathrm{H}} \triangleq \mu\alpha.\langle [\![M]\!]^{\lambda}_{\mathrm{H}} \,|\, [\![P]\!]^{\lambda}_{\mathrm{H}} \cdot \alpha\rangle \to^{\mathrm{N}*}_{\overline{\lambda}} (IH)\ \mu\alpha.\langle [\![N]\!]^{\lambda}_{\mathrm{H}} \,|\, [\![P]\!]^{\lambda}_{\mathrm{H}} \cdot \alpha\rangle \triangleq [\![NP]\!]^{\lambda}_{\mathrm{H}}$.

*iii)* $(\lambda x.M)V \to_{\mathrm{v}} M\{V/x\}$: As in part (*i*) above; notice that then $N \equiv V$, and the $\tilde{\mu}$-reduction step is permitted.

$M \to_{\mathrm{v}} N \Rightarrow MP \to_{\mathrm{v}} NP$: $[\![MP]\!]^{\lambda}_{\mathrm{H}} \triangleq \mu\alpha.\langle [\![M]\!]^{\lambda}_{\mathrm{H}} \,|\, [\![P]\!]^{\lambda}_{\mathrm{H}} \cdot \alpha\rangle \to^{\mathrm{v}*}_{\overline{\lambda}} (IH)\ \mu\alpha.\langle [\![N]\!]^{\lambda}_{\mathrm{H}} \,|\, [\![P]\!]^{\lambda}_{\mathrm{H}} \cdot \alpha\rangle \triangleq [\![NP]\!]^{\lambda}_{\mathrm{H}}$.

$M \to_{\mathrm{v}} N \Rightarrow VM \to_{\mathrm{v}} VN$: $[\![VM]\!]^{\lambda}_{\mathrm{H}} \triangleq \mu\alpha.\langle [\![V]\!]^{\lambda}_{\mathrm{H}} \,|\, [\![M]\!]^{\lambda}_{\mathrm{H}} \cdot \alpha\rangle \to^{\mathrm{v}*}_{\overline{\lambda}} (IH)\ \mu\alpha.\langle [\![V]\!]^{\lambda}_{\mathrm{H}} \,|\, [\![N]\!]^{\lambda}_{\mathrm{H}} \cdot \alpha\rangle \triangleq [\![VN]\!]^{\lambda}_{\mathrm{H}}$. □

This interpretation corresponds to running the $\lambda$-calculus in a stack-based CBN abstract machine (like Krivine's machine [13]): 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 4.8* In $\overline{\lambda}\mu\tilde{\mu}$ we express the interaction between a program (term) and its environment via commands. Although there is no notion of application, $\overline{\lambda}\mu\tilde{\mu}$ sees $[\![MN_1\cdots N_n]\!]^{\lambda}_{\mathrm{H}}$ as running $[\![M]\!]^{\lambda}_{\mathrm{H}}$ in the environment that offers the terms $[\![N_1]\!]^{\lambda}_{\mathrm{H}}, \ldots, [\![N_n]\!]^{\lambda}_{\mathrm{H}}$ in sequence. To understand this, first notice that

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

so it is easy to verify that

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

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

# 5 Mapping CDR to $\overline{\lambda}\mu\tilde{\mu}$

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

**Definition 5.1** (Direct interpretation) The interpretation $[\![\cdot]\!]^{\mathrm{D}}$ of CDR values and computations into $\overline{\lambda}\mu\tilde{\mu}$-terms is defined as:

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

Notice that this interpretation does without forcing, thunking, and return, but just simply puts all the sub-terms that are not computations on the stack; since nothing will be allowed to run on the stack in CBN reduction, computation there is halted automatically. Since in CDR thunked terms can only appear as arguments in an application, by the interpretation these are all placed on the stack in the $\overline{\lambda}\mu\tilde{\mu}$-term, where under CBN reduction is not permitted, so no syntactic marker is necessary to block computation.

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 5.2* $[\![M\lceil V/x\rfloor]\!]^{\mathrm{D}} = [\![M]\!]^{\mathrm{D}}\{[\![V]\!]^{\mathrm{D}}/x\}$.

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

$x\lceil V/x\rfloor = V$: We have two cases:

$V = y$: $[\![x\lceil y/x\rfloor]\!]^{\mathrm{D}} = [\![y]\!]^{\mathrm{D}} = y = x\{y/x\} = [\![x]\!]^{\mathrm{D}}\{[\![y]\!]^{\mathrm{D}}/x\}$.

$V = \{N\}$: $[\![x\lceil\{N\}/x\rfloor]\!]^{\mathrm{D}} = [\![\{N\}]\!]^{\mathrm{D}} = [\![N]\!]^{\mathrm{D}} = x\{[\![N]\!]^{\mathrm{D}}/x\} = [\![x]\!]^{\mathrm{D}}\{[\![\{N\}]\!]^{\mathrm{D}}/x\}$.

$z\lceil V/x\rfloor = z$: $[\![z\lceil V/x\rfloor]\!]^{\mathrm{D}} = [\![z]\!]^{\mathrm{D}} = z = z\{[\![V]\!]^{\mathrm{D}}/x\} = [\![z]\!]^{\mathrm{D}}\{[\![V]\!]^{\mathrm{D}}/x\}$.

$x!\lceil y/x\rfloor = y!$: $[\![x!\lceil y/x\rfloor]\!]^{\mathrm{D}} = [\![y!]\!]^{\mathrm{D}} = y = x\{y/x\} = [\![x!]\!]^{\mathrm{D}}\{[\![y]\!]^{\mathrm{D}}/x\}$.

$x!\lceil\{N\}/x\rfloor = N : [\![x!\lceil\{N\}/x\rfloor]\!]^{\mathrm{D}} = [\![N]\!]^{\mathrm{D}} = x\{[\![N]\!]^{\mathrm{D}}/x\} = [\![x!]\!]^{\mathrm{D}}\{[\![\{N\}]\!]^{\mathrm{D}}/x\}.$
$z!\lceil y/x\rfloor = z! : [\![z!\lceil y/x\rfloor]\!]^{\mathrm{D}} = [\![z!]\!]^{\mathrm{D}} = z = z\{[\![y]\!]^{\mathrm{D}}/x\} = [\![z!]\!]^{\mathrm{D}}\{[\![y]\!]^{\mathrm{D}}/x\}.$
$z!\lceil\{N\}/x\rfloor = z! : [\![z!\lceil\{N\}/x\rfloor]\!]^{\mathrm{D}} = [\![z!]\!]^{\mathrm{D}} = z = z\{[\![\{N\}]\!]^{\mathrm{D}}/x\} = [\![z!]\!]^{\mathrm{D}}\{[\![\{N\}]\!]^{\mathrm{D}}/x\}.$ □

Using this result, we can now show that reduction in CDR is preserved by CBN reduction in $\overline{\lambda}\mu\tilde{\mu}$ under the interpretation $[\![\cdot]\!]^{\mathrm{D}}$.

**Theorem 5.3** *If $M \to_{\mathrm{D}} N$, then $[\![M]\!]^{\mathrm{D}} \to_{\overline{\lambda}}^{\mathrm{N}*} [\![N]\!]^{\mathrm{D}}$.*

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

$(\lambda x.M)V \to M\lceil V/x\rfloor$: $[\![(\lambda x.M)\,V]\!]^{\mathrm{D}} \stackrel{\Delta}{=} \mu\alpha.\langle\lambda x.[\![M]\!]^{\mathrm{D}} \,|\, [\![V]\!]^{\mathrm{D}}\cdot\alpha\rangle \to_{\mathrm{P}}^{\mathrm{N}} (\lambda)$
$\mu\alpha.\langle [\![V]\!]^{\mathrm{D}} \,|\, \tilde{\mu}x.\langle [\![M]\!]^{\mathrm{D}} \,|\, \alpha\rangle\rangle \to_{\mathrm{P}}^{\mathrm{N}} (\tilde{\mu})\ \mu\alpha.\langle [\![M]\!]^{\mathrm{D}}\{[\![V]\!]^{\mathrm{D}}/x\} \,|\, \alpha\rangle \to_{\mathrm{P}}^{\mathrm{N}} (\eta\mu)$
$[\![M]\!]^{\mathrm{D}}\{[\![V]\!]^{\mathrm{D}}/x\} = (5.2)\ [\![M\lceil V/x\rfloor]\!]^{\mathrm{D}}$

$x := ret\,V; N \to N\lceil V/x\rfloor$: $[\![x := ret\,V; N]\!]^{\mathrm{D}} \stackrel{\Delta}{=} \mu\alpha.\langle [\![V]\!]^{\mathrm{D}} \,|\, \tilde{\mu}x.\langle [\![N]\!]^{\mathrm{D}} \,|\, \alpha\rangle\rangle \to_{\mathrm{P}}^{\mathrm{N}} (\tilde{\mu})$
$\mu\alpha.\langle [\![N]\!]^{\mathrm{D}}\{[\![V]\!]^{\mathrm{D}}/x\} \,|\, \alpha\rangle \to_{\mathrm{P}}^{\mathrm{N}} (\eta\mu)\ [\![N]\!]^{\mathrm{D}}\{[\![V]\!]^{\mathrm{D}}/x\} = (5.2)\ [\![N\lceil V/x\rfloor]\!]^{\mathrm{D}}$

$M \to_{\mathrm{P}} N \Rightarrow MV \to_{\mathrm{P}} NV$: $[\![MV]\!]^{\mathrm{D}} \stackrel{\Delta}{=} \mu\alpha.\langle [\![M]\!]^{\mathrm{D}} \,|\, [\![V]\!]^{\mathrm{D}}\cdot\alpha\rangle \to_{\overline{\lambda}}^{\mathrm{N}*} (IH)\ \mu\alpha.\langle [\![N]\!]^{\mathrm{D}} \,|\, [\![V]\!]^{\mathrm{D}}\cdot\alpha\rangle = [\![NV]\!]^{\mathrm{D}}$

$M \to_{\mathrm{P}} N \Rightarrow x := M; P \to x := N; P$: $[\![x := M; P]\!]^{\mathrm{D}} \stackrel{\Delta}{=} \mu\alpha.\langle [\![M]\!]^{\mathrm{D}} \,|\, \tilde{\mu}x.\langle [\![P]\!]^{\mathrm{D}} \,|\, \alpha\rangle\rangle \to_{\overline{\lambda}}^{\mathrm{N}*} (IH)$
$\mu\alpha.\langle [\![N]\!]^{\mathrm{D}} \,|\, \tilde{\mu}x.\langle [\![P]\!]^{\mathrm{D}} \,|\, \alpha\rangle\rangle = [\![x := N; P]\!]^{\mathrm{D}}$ □

Notice the use of CBN reduction: this stresses again that reduction in CDR (and thereby also in CBPV) is essentially CBN; since all redexes are unique and reduction is deterministic, we have:

**Theorem 5.4** (FULL ABSTRACTION) *If $[\![M]\!]^{\mathrm{D}} \to_{\overline{\lambda}}^{\mathrm{N}*} Q$, then there exists $N \in$ CDR such that $M \to_{\mathrm{D}} N$, and $Q \to_{\overline{\lambda}}^{\mathrm{N}*} [\![N]\!]^{\mathrm{D}}$.*

This result shows that $\overline{\lambda}\mu\tilde{\mu}$ is an ideal calculus to implement CDR.

Using the 'inverse' of the type interpretation from Def. 3.15, we can also show that type assignment is preserved under the interpretations.

**Definition 5.5** The type interpretation $[\![\cdot]\!]^{\mathrm{D}}$ is defined as:

$$\begin{aligned} [\![\varphi]\!]^{\mathrm{D}} &= \varphi \\ [\![U\mathsf{A}]\!]^{\mathrm{D}} &= [\![\mathsf{A}]\!]^{\mathrm{D}} \\ [\![A \to \mathsf{B}]\!]^{\mathrm{D}} &= [\![A]\!]^{\mathrm{D}} \to [\![\mathsf{B}]\!]^{\mathrm{D}} \\ [\![F A]\!]^{\mathrm{D}} &= [\![A]\!]^{\mathrm{D}} \end{aligned}$$

and the environment interpretation as: $[\![\Gamma]\!]^{\mathrm{D}} = \{\, x{:}[\![A]\!]^{\mathrm{D}} \mid x{:}A \in \Gamma \,\}$.

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 5.6** *i) If $\Gamma \vdash_{\mathrm{V}} V : A$, then $[\![\Gamma]\!]^{\mathrm{D}} \vdash [\![V]\!]^{\mathrm{D}} : [\![A]\!]^{\mathrm{D}} \mid$ .*
*ii) If $\Gamma \vdash_{\mathrm{C}} M : \mathsf{A}$, then $[\![\Gamma]\!]^{\mathrm{D}} \vdash [\![M]\!]^{\mathrm{D}} : [\![\mathsf{A}]\!]^{\mathrm{D}} \mid$ .*

*Proof:* By induction on definition of type assignment for CDR terms.

(*axiom*): Then $V \equiv x$, and $\Gamma = \Gamma', x{:}A$; since $[\![x]\!]^{\mathrm{D}} \stackrel{\Delta}{=} x$, and $[\![\Gamma]\!]^{\mathrm{D}} = [\![\Gamma']\!]^{\mathrm{D}}, x{:}[\![A]\!]^{\mathrm{D}}$, by rule $(AxR)$ also $[\![\Gamma]\!]^{\mathrm{D}} \vdash [\![v]\!]^{\mathrm{D}} : [\![A]\!]^{\mathrm{D}} \mid$ .

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

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

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

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

(*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, $[\![\Gamma]\!]^{\mathrm{D}} \vdash [\![N]\!]^{\mathrm{D}} : [\![B \to \mathsf{A}]\!]^{\mathrm{D}} \mid$ and $[\![\Gamma]\!]^{\mathrm{D}} \vdash [\![V]\!]^{\mathrm{D}} : [\![B]\!]^{\mathrm{D}} \mid$ . We can construct (with $\Gamma' = [\![\Gamma]\!]^{\mathrm{D}}, x{:}U(B \to \mathsf{A}), y{:}B$):

$$\dfrac{\dfrac{\dfrac{\boxed{\phantom{[\![\Gamma]\!]^{\mathrm{D}} \vdash [\![N]\!]^{\mathrm{D}} : [\![B \to \mathsf{A}]\!]^{\mathrm{D}} \mid}}}{[\![\Gamma]\!]^{\mathrm{D}} \vdash [\![N]\!]^{\mathrm{D}} : [\![B \to \mathsf{A}]\!]^{\mathrm{D}} \mid \alpha{:}[\![\mathsf{A}]\!]^{\mathrm{D}}}\ (Wk) \quad \dfrac{\dfrac{\boxed{\phantom{[\![\Gamma]\!]^{\mathrm{D}} \vdash [\![V]\!]^{\mathrm{D}} : [\![B]\!]^{\mathrm{D}} \mid}}}{[\![\Gamma]\!]^{\mathrm{D}} \vdash [\![V]\!]^{\mathrm{D}} : [\![B]\!]^{\mathrm{D}} \mid \alpha{:}[\![\mathsf{A}]\!]^{\mathrm{D}}}\ (Wk) \quad \dfrac{}{[\![\Gamma]\!]^{\mathrm{D}} \mid \alpha : [\![\mathsf{A}]\!]^{\mathrm{D}} \vdash \alpha{:}[\![\mathsf{A}]\!]^{\mathrm{D}}}\ (AxL)}{[\![\Gamma]\!]^{\mathrm{D}} \mid [\![V]\!]^{\mathrm{D}} \cdot \alpha : [\![B \to \mathsf{A}]\!]^{\mathrm{D}} \vdash \alpha{:}[\![\mathsf{A}]\!]^{\mathrm{D}}}\ (\to L)}{\langle [\![M]\!]^{\mathrm{D}} \mid [\![V]\!]^{\mathrm{D}} \cdot \alpha \rangle : [\![\Gamma]\!]^{\mathrm{D}} \vdash \alpha{:}[\![\mathsf{A}]\!]^{\mathrm{D}}}\ (cut)}{[\![\Gamma]\!]^{\mathrm{D}} \vdash \mu\alpha.\langle [\![N]\!]^{\mathrm{D}} \mid [\![V]\!]^{\mathrm{D}} \cdot \alpha \rangle : [\![\mathsf{A}]\!]^{\mathrm{D}} \mid}\ (\mu)$$

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

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

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

(*seq*): 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 $[\![\Gamma]\!]^{\mathrm{D}} \vdash [\![P]\!]^{\mathrm{D}} : [\![\mathsf{F}B]\!]^{\mathrm{D}} \mid$ and $[\![\Gamma, x{:}B]\!]^{\mathrm{D}} \vdash [\![Q]\!]^{\mathrm{D}} : [\![\mathsf{A}]\!]^{\mathrm{D}} \mid$ . Notice that $[\![\Gamma, x{:}B]\!]^{\mathrm{D}} \triangleq [\![\Gamma]\!]^{\mathrm{D}}, x{:}[\![B]\!]^{\mathrm{D}}$, $[\![\mathsf{F}B]\!]^{\mathrm{D}} = \triangleq [\![B]\!]^{\mathrm{D}}$, and $[\![\mathsf{A}]\!]^{\mathrm{D}} = \triangleq [\![A]\!]^{\mathrm{D}}$.

$$\dfrac{\dfrac{\boxed{\phantom{[\![\Gamma]\!]^{\mathrm{D}} \vdash [\![P]\!]^{\mathrm{D}} : [\![B]\!]^{\mathrm{D}} \mid}}}{[\![\Gamma]\!]^{\mathrm{D}} \vdash [\![P]\!]^{\mathrm{D}} : [\![B]\!]^{\mathrm{D}} \mid \alpha{:}[\![\mathsf{A}]\!]^{\mathrm{D}}}\ (Wk) \quad \dfrac{\dfrac{\boxed{\phantom{[\![\Gamma]\!]^{\mathrm{D}}, x{:}[\![B]\!]^{\mathrm{D}} \vdash [\![Q]\!]^{\mathrm{D}} : [\![\mathsf{A}]\!]^{\mathrm{D}} \mid}}}{[\![\Gamma]\!]^{\mathrm{D}}, x{:}[\![B]\!]^{\mathrm{D}} \vdash [\![Q]\!]^{\mathrm{D}} : [\![\mathsf{A}]\!]^{\mathrm{D}} \mid \alpha{:}[\![\mathsf{A}]\!]^{\mathrm{D}}}\ (Wk) \quad \vdots \quad \dfrac{}{[\![\Gamma]\!]^{\mathrm{D}}, x{:}[\![B]\!]^{\mathrm{D}} \mid \alpha : [\![\mathsf{A}]\!]^{\mathrm{D}} \vdash \alpha{:}[\![\mathsf{A}]\!]^{\mathrm{D}}}\ (AxL)}{\langle [\![Q]\!]^{\mathrm{D}} \mid \alpha \rangle : [\![\Gamma]\!]^{\mathrm{D}}, x{:}[\![B]\!]^{\mathrm{D}} \vdash \alpha{:}[\![\mathsf{A}]\!]^{\mathrm{D}}}\ (cut)}{[\![\Gamma]\!]^{\mathrm{D}} \mid \tilde{\mu}x.\langle [\![Q]\!]^{\mathrm{D}} \mid \alpha \rangle : [\![B]\!]^{\mathrm{D}} \vdash \alpha{:}[\![\mathsf{A}]\!]^{\mathrm{D}}}\ (\tilde{\mu})}{\langle [\![P]\!]^{\mathrm{D}} \mid \tilde{\mu}x.\langle [\![Q]\!]^{\mathrm{D}} \mid \alpha \rangle \rangle : [\![\Gamma]\!]^{\mathrm{D}} \vdash \alpha{:}[\![\mathsf{A}]\!]^{\mathrm{D}}}\ (cut)}{[\![\Gamma]\!]^{\mathrm{D}} \vdash \mu\alpha.\langle [\![P]\!]^{\mathrm{D}} \mid \rho x.\langle [\![Q]\!]^{\mathrm{D}} \mid \alpha \rangle \rangle : [\![\mathsf{A}]\!]^{\mathrm{D}} \mid}\ (\mu)$$

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

We can also show that reduction in $\textsc{ck}^{\mathrm{D}}$ is respected by reduction in $\overline{\lambda}\mu\tilde{\mu}$.

**Definition 5.7** The interpretation of evaluation stacks and configurations in $\textsc{ck}^{\mathrm{D}}$ into $\overline{\lambda}\mu\tilde{\mu}$ is defined through:

$$\begin{aligned}
[\![\epsilon]\!]^{\mathrm{D}}_{\textsc{ck}}\alpha &= \alpha & [\![\langle MV \mid S \rangle]\!]^{\mathrm{D}}_{\textsc{ck}} &= \mu\alpha.\langle [\![M]\!]^{\mathrm{D}} \mid [\![V]\!]^{\mathrm{D}} \cdot [\![S]\!]^{\mathrm{D}}_{\textsc{ck}}\alpha \rangle \\
[\![V : S]\!]^{\mathrm{D}}_{\textsc{ck}}\alpha &= [\![V]\!]^{\mathrm{D}} \cdot [\![S]\!]^{\mathrm{D}}_{\textsc{ck}}\alpha & [\![\langle x := M; N \mid S \rangle]\!]^{\mathrm{D}}_{\textsc{ck}} &= \mu\alpha.\langle [\![M]\!]^{\mathrm{D}} \mid \tilde{\mu}x.\langle [\![N]\!]^{\mathrm{D}} \mid [\![S]\!]^{\textsc{ck}}_{\alpha} \rangle \rangle \\
[\![x := [\,]; N : S]\!]^{\mathrm{D}}_{\textsc{ck}}\alpha &= \tilde{\mu}x.\langle [\![N]\!]^{\mathrm{D}} \mid [\![S]\!]^{\mathrm{D}}_{\textsc{ck}}\alpha \rangle & [\![\langle M \mid S \rangle]\!]^{\mathrm{D}}_{\textsc{ck}} &= \mu\alpha.\langle [\![M]\!]^{\mathrm{D}} \mid [\![S]\!]^{\mathrm{D}}_{\textsc{ck}}\alpha \rangle \ (otherwise)
\end{aligned}$$

It was this similarity between stacks in $\textsc{ck}$ and $\overline{\lambda}\mu\tilde{\mu}$ that was the inspiration for this paper.

We can now show that this interpretation respects $\to^{\mathrm{D}}_{\textsc{ck}}$-reductions.

**Theorem 5.8** *If* $\langle M \mid S\rangle \to^{\mathrm{D}}_{\mathrm{CK}} \langle N \mid S\rangle$, *then* $[\![\langle M \mid S\rangle]\!]^{\mathrm{D}}_{\mathrm{CK}} \to^{\mathrm{N}*}_{\overline{\lambda}} [\![\langle N \mid S\rangle]\!]^{\mathrm{D}}_{\mathrm{CK}}$.

*Proof:* $\langle \lambda x.M \mid V : S\rangle \to^{\mathrm{D}}_{\mathrm{CK}} \langle M\lceil V/x\rfloor \mid S\rangle$: $[\![\langle \lambda x.M \mid V : S\rangle]\!]^{\mathrm{D}}_{\mathrm{CK}} \stackrel{\Delta}{=}$
$\mu\alpha.\langle \lambda x.[\![M]\!]^{\mathrm{D}} \mid [\![V]\!]^{\mathrm{D}} \cdot [\![S]\!]^{\mathrm{D}}_{\mathrm{CK}}\alpha\rangle \to^{\mathrm{N}}_{\mathrm{P}} \mu\alpha.\langle [\![V]\!]^{\mathrm{D}} \mid \tilde{\mu}x.\langle [\![M]\!]^{\mathrm{D}} \mid [\![S]\!]^{\mathrm{D}}_{\mathrm{CK}}\alpha\rangle\rangle \to^{\mathrm{N}}_{\mathrm{P}}$
$\mu\alpha.\langle [\![M]\!]^{\mathrm{D}}\{[\![V]\!]^{\mathrm{D}}/x\} \mid [\![S]\!]^{\mathrm{D}}_{\mathrm{CK}}\alpha\rangle = (5.2)\ \mu\alpha.\langle [\![M\lceil V/x\rfloor]\!]^{\mathrm{D}} \mid [\![S]\!]^{\mathrm{D}}_{\mathrm{CK}}\alpha\rangle \stackrel{\Delta}{=} [\![\langle M\lceil V/x\rfloor \mid S\rangle]\!]^{\mathrm{D}}_{\mathrm{CK}}$

$\langle MV \mid S\rangle \to^{\mathrm{D}}_{\mathrm{CK}} \langle M \mid V : S\rangle$: $[\![\langle MV \mid S\rangle]\!]^{\mathrm{D}}_{\mathrm{CK}} \stackrel{\Delta}{=} \mu\alpha.\langle [\![M]\!]^{\mathrm{D}} \mid [\![V]\!]^{\mathrm{D}} \cdot [\![S]\!]^{\mathrm{D}}_{\mathrm{CK}}\alpha\rangle \stackrel{\Delta}{=} [\![\langle M \mid V : S\rangle]\!]^{\mathrm{D}}_{\mathrm{CK}}$

$\langle x := M; N \mid S\rangle \to^{\mathrm{D}}_{\mathrm{CK}} \langle M \mid x := [\,]; N : S\rangle$: $[\![\langle x := M; N \mid S\rangle]\!]^{\mathrm{D}}_{\mathrm{CK}} \stackrel{\Delta}{=}$
$\mu\alpha.\langle [\![M]\!]^{\mathrm{D}} \mid \tilde{\mu}x.\langle [\![N]\!]^{\mathrm{D}} \mid [\![S]\!]^{\mathrm{D}}_{\mathrm{CK}}\alpha\rangle\rangle \stackrel{\Delta}{=} \mu\alpha.\langle [\![M]\!]^{\mathrm{D}} \mid [\![x := [\,]; N : S]\!]^{\mathrm{D}}_{\mathrm{CK}}\alpha\rangle \stackrel{\Delta}{=} [\![\langle M \mid x := [\,]; N : S\rangle]\!]^{\mathrm{D}}_{\mathrm{CK}}$

$\langle ret\, V \mid x := [\,]; M : S\rangle \to^{\mathrm{D}}_{\mathrm{CK}} \langle M\lceil V/x\rfloor \mid S\rangle$: $[\![\langle ret\, V \mid x := [\,]; M : S\rangle]\!]^{\mathrm{D}}_{\mathrm{CK}} \stackrel{\Delta}{=}$
$\mu\alpha.\langle [\![V]\!]^{\mathrm{D}} \mid \tilde{\mu}x.\langle [\![M]\!]^{\mathrm{D}} \mid [\![S]\!]^{\mathrm{D}}_{\mathrm{CK}}\alpha\rangle\rangle \to^{\mathrm{N}}_{\mathrm{P}} \mu\alpha.\langle [\![M]\!]^{\mathrm{D}}\{[\![V]\!]^{\mathrm{D}}/x\} \mid [\![S]\!]^{\mathrm{D}}_{\mathrm{CK}}\alpha\rangle \stackrel{\Delta}{=} (5.2)$
$[\![\langle M\lceil V/x\rfloor \mid S\rangle]\!]^{\mathrm{D}}_{\mathrm{CK}}$ □

## Conclusion and Future work

We have shown that, although partially presented for that purpose, Levy's CBPV calculus is not really suitable to represent the CBN or CBV reduction of the $\lambda$-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 CDR, 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 single step CBN or CBV reduction of the $\lambda$-calculus, as well as preserve typeability. Moreover, we have defined a mapping of CDR into Curien and Herbelin's $\overline{\lambda}\mu\tilde{\mu}$, and showed that reduction in CDR can be successfully modeled in the CBN partition of $\overline{\lambda}\mu\tilde{\mu}$, as well as that typeability is preserved.

In future work, we will investigate the role of $ret\,\cdot$, as well as define a notion of type assignment for $\overline{\lambda}\mu\tilde{\mu}$ using CBPV types that can better express the relation between CDR and $\overline{\lambda}\mu\tilde{\mu}$. We will also look at the role of the assignment term $x := M; N$, and see if it could be used to define CBV reduction for CBPV.

Although all proofs in this paper follow by straightforward induction, and we have no reason to assume that we missed some detail, we intend to haul everything through a theorem prover, just to be sure.

## 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] A. Church and J.B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39:472–482, 1936.

[4] 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.

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

[6] T. Ehrhard. Call-By-Push-Value from a Linear Logic Point of View. In P. Thiemann, editor, *Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9632 of *Lecture Notes in Computer Science*, pages 202–228. Springer, 2016.

[7] T. Ehrhard and G. Guerrieri. The Bang Calculus: an untyped lambda-calculus generalizing call-by-name and call-by-value. In J. Cheney and G. Vidal, editors, *Proceedings of the 18th International Symposium on Principles and Practice of Declarative Programming, Edinburgh, United Kingdom, September 5-7, 2016*, pages 174–187. ACM, 2016.

[8] Y. Forster, S. Schäfer, S. Spies, and K. 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.

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

[10] J.-Y Girard. Linear Logic. *Theoretical Computer Science*, 50:1–102, 1987.

[11] 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.

[12] 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.

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

[14] 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.

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

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

[17] E. Moggi. Notions of Computation and Monads. *Information and Computation*, 93:55–92, 1991.

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

[19] G.D. Plotkin. Call-by-name, call-by-value and the lambda-calculus. *Theoretical Computer Science*, 1:125–159, 1975.

[20] 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.

[21] 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.