Open Call-by-Value

Beniamino Accattoli, Giulio Guerrieri

Open Call-by-Value

Beniamino Accattoli[1] and Giulio Guerrieri[2]

[1] INRIA, UMR 7161, LIX, École Polytechnique, beniamino.accattoli@inria.fr
[2] Aix Marseille Univ, CNRS, Centrale Marseille, I2M, Marseille, France, giulio.guerrieri@univ-amu.fr

Abstract. The elegant theory of the call-by-value lambda-calculus relies on weak evaluation and closed terms, that are natural hypotheses in the study of programming languages. To model proof assistants, however, strong evaluation and open terms are required, and it is well known that the operational semantics of call-by-value becomes problematic in this case. Here we study the intermediate setting—that we call Open Call-by-Value—of weak evaluation with open terms, on top of which Grégoire and Leroy designed the abstract machine of Coq. Various calculi for Open Call-by-Value already exist, each one with its pros and cons. This paper presents a detailed comparative study of the operational semantics of four of them, coming from different areas such as the study of abstract machines, denotational semantics, linear logic proof nets, and sequent calculus. We show that these calculi are all equivalent from a termination point of view, justifying the slogan Open Call-by-Value.

1 Introduction

Plotkin's call-by-value λ-calculus [26] is at the heart of programming languages such as OCaml and proof assistants such as Coq. In the study of programming languages, call-by-value (CBV) evaluation is usually *weak*, *i.e.* it does not reduce under abstractions, and terms are assumed to be *closed*. These constraints give rise to a beautiful theory—let us call it *Closed CBV*—having the following *harmony property*, that relates rewriting and normal forms:

Closed normal forms are values (and values are normal forms)

where *values* are variables and abstractions. Harmony expresses a form of internal completeness with respect to unconstrained β-reduction: the restriction to CBV β-reduction (referred to as β_v*-reduction*, according to which a β-redex can be fired only when the argument is a value) has an impact on the order in which redexes are evaluated, but evaluation never gets stuck, as every β-redex will eventually become a β_v-redex and be fired, unless evaluation diverges.

It often happens, however, that one needs to go beyond the perfect setting of Closed CBV by considering *Strong CBV*, where reduction under abstractions is allowed and terms may be open, or the intermediate setting of *Open CBV*, where evaluation is weak but terms are not necessarily closed. The need arises, most notably, when trying to describe the implementation model of Coq [13], but also

from other motivations, as denotational semantics [25,28,4,8], monad and CPS translations and the associated equational theories [21,29,30,12,16], bisimulations [18], partial evaluation [17], linear logic proof nets [2], or cost models [1].

Naïve Open CBV. In call-by-name (CBN) turning to open terms or strong evaluation is harmless because CBN does not impose any special form to the arguments of β-redexes. On the contrary, turning to Open or Strong CBV is delicate. If one simply considers Plotkin's weak β_v-reduction on open terms—let us call it *Naïve Open CBV*—then harmony does no longer hold, as there are open β-normal forms that are not values, *e.g.* xx, $x(\lambda y.y)$, $x(yz)$ or xyz. As a consequence, there are *stuck β-redexes* such as $(\lambda y.t)(xx)$, *i.e.* β-redexes that will never be fired because their argument is normal, but it is not a value, nor will it ever become one. Such stuck β-redexes are a disease typical of (Naïve) Open CBV, but they spread to Strong CBV as well (also in the closed case), because evaluating under abstraction forces to deal with locally open terms: *e.g.* the variable x is locally open with respect to $(\lambda y.t)(xx)$ in $s = \lambda x.((\lambda y.t)(xx))$.

The real issue with stuck β-redexes is that they prevent the creation of other redexes, and provide *premature* β_v-normal forms. The issue is serious, as it can affect termination, and thus impact on notions of observational equivalence. Let $\delta \coloneqq \lambda x.(xx)$. The problem is exemplified by the terms t and u in Eq. (1) below.

$$t \coloneqq ((\lambda y.\delta)(zz))\delta \qquad\qquad u \coloneqq \delta((\lambda y.\delta)(zz)) \tag{1}$$

In Naïve Open CBV, t and u are premature β_v-normal forms because they both have a stuck β-redex forbidding evaluation to keep going, while one would expect them to behave like the divergent term $\Omega \coloneqq \delta\delta$ (see [25,28,4,2,8,15] and pp. 7-12).

Open CBV. In his seminal work, Plotkin already pointed out an asymmetry between CBN and CBV: his CPS translation is sound and complete for CBN, but only sound for CBV. This fact led to a number of studies about monad, CPS, and logical translations [21,29,30,20,12,16] that introduced many proposals of improved calculi for CBV. Starting with the seminal work of Paolini and Ronchi Della Rocca [25,23,28], the dissonance between open terms and CBV has been repeatedly pointed out and studied *per se* via various calculi [13,4,2,8,15,14,1]. A further point of view on CBV comes from the computational interpretation of sequent calculus due to Curien and Herbelin [9]. An important point is that the focus of most of these works is on Strong CBV.

These solutions inevitably extend β_v-reduction with some other rewriting rule(s) or constructor (as `let`-expressions) to deal with stuck β-redexes, or even go as far as changing the applicative structure of terms, as in the sequent calculus approach. They arise from different perspectives and each one has its pros and cons. By design, these calculi (when looked at in the context of Open CBV) are never observationally equivalent to Naïve Open CBV, as they all manage to (re)move stuck β-redexes and may diverge when Naïve Open CBV is instead stuck. Each one of these calculi, however, has its own notion of evaluation and normal form, and their mutual relationships are not evident.

The aim of this paper is to draw the attention of the community on Open CBV. We believe that it is somewhat deceiving that the mainstream operational theory of CBV, however elegant, has to rely on closed terms, because it restricts the modularity of the framework, and raises the suspicion that the true essence of CBV has yet to be found. There is a real gap, indeed, between Closed and Strong CBV, as Strong CBV cannot be seen as an iteration of Closed CBV under abstractions because such an iteration has to deal with open terms. To improve the implementation of Coq [13], Grégoire and Leroy see Strong CBV as the iteration of the intermediate case of Open CBV, but they do not explore its theory. Here we exalt their point of view, providing a thorough operational study of Open CBV. We insist on Open CBV rather than Strong CBV because:

1. Stuck β-redexes and premature β_v-normal forms already affect Open CBV;
2. Open CBV has a simpler rewriting theory than Strong CBV;
3. Our previous studies of Strong CBV in [4] and [8] naturally organized themselves as properties of Open CBV that were lifted to Strong CBV by a simple iteration under abstractions.

Our contributions are along two axes:

1. *Termination Equivalence of the Proposals*: we show that the proposed generalizations of Naïve Open CBV are all equivalent, in the sense that they have exactly the same sets of normalizing and diverging terms. So, *there is just one notion of Open CBV*, independently of its specific syntactic incarnation.
2. *Quantitative Analyses and Cost Models*: the termination results are complemented with quantitative analyses establishing precise relationships between the number of steps needed to evaluate a given term in the various calculi. In particular, we relate the cost models of the various proposals.

The Fab Four. We focus on four proposals for Open CBV, as other solutions, *e.g.* Moggi's [21] or Herbelin and Zimmerman's [16], are already known to be equivalent to these ones (see the end of Sect. 2):

1. *The Fireball Calculus* λ_{fire}, that extends values to *fireballs* by adding so-called *inert terms* in order to restore harmony—it was introduced without a name by Paolini and Ronchi Della Rocca [25,28], then rediscovered independently first by Leroy and Grégoire [13] to improve the implementation of Coq, and then by Accattoli and Sacerdoti Coen [1] to study cost models;
2. *The Value Substitution Calculus* λ_{vsub}, coming from the linear logic interpretation of CBV and using explicit substitutions and contextual rewriting rules to circumvent stuck β-redexes—it was introduced by Accattoli and Paolini [4] and it is a graph-free presentation of proof nets for the CBV λ-calculus [2];
3. *The Shuffling Calculus* λ_{shuf}, that has rules to shuffle constructors, similar to Regnier's σ-rules for CBN [27], as an alternative to explicit substitutions—it was introduced by Carraro and Guerrieri [8] (and further analyzed in [15,14]) to study the adequacy of Open/Strong CBV with respect to denotational semantics related to linear logic.
4. *The Value Sequent Calculus* λ_{vseq}, *i.e.* the intuitionistic fragment of Curien and Herbelin's $\bar{\lambda}\tilde{\mu}$-calculus [9], that is a CBV calculus for classical logic providing a

computational interpretation of sequent calculus rather than natural deduction (in turn a fragment of the $\overline{\lambda}\mu\tilde{\mu}$-calculus [9], further studied in *e.g.* [6,10]).

A Robust Cost Model for Open CBV. The number of β_v-steps is the canonical time cost model of Closed CBV, as first proved by Blelloch and Greiner [7,31,11]. In [1], Accattoli and Sacerdoti Coen generalized this result: the number of steps in λ_{fire} is a reasonable cost model for Open CBV. Here we show that the number of steps in λ_{vsub} and λ_{vseq} are *linearly* related to the steps in λ_{fire}, thus providing reasonable cost models for these incarnations of Open CBV. As a consequence, complexity analyses can now be smoothly transferred between λ_{fire}, λ_{vsub}, and λ_{vseq}. Said differently, our results guarantee that the number of steps is a *robust* cost model for Open CBV, in the sense that it does not depend on the chosen incarnation. For λ_{shuf} we obtain a similar but strictly weaker result, due to some structural difficulties suggesting that λ_{shuf} is less apt to complexity analyses.

On the Value of The Paper. While the equivalences showed here are new, they might not be terribly surprising. Nonetheless, we think they are interesting, for the following reasons:

1. *Quantitative Relationships*: λ-calculi are usually related only *qualitatively*, while our relationships are *quantitative* and thus stronger: not only we show simulations, but we also relate the number of steps.
2. *Uniform View*: we provide a new uniform view on a known problem, that will hopefully avoid further proliferations of CBV calculi for open/strong settings.
3. *Expected but Non-Trivial*: while the equivalences are more or less expected, establishing them is informative, because it forces to reformulate and connect concepts among the different settings, and often tricky.
4. *Simple Rewriting Theory*: the relationships between the systems are developed using basic rewriting concepts. The technical development is simple, according to the best tradition of the CBV λ-calculus, and yet it provides a sharp and detailed decomposition of Open CBV evaluation.
5. *Connecting Different Worlds*: while λ_{fire} is related to Coq and implementations, λ_{vsub} and λ_{shuf} have a linear logic background, and λ_{vseq} is rooted in sequent calculus. With respect to linear logic, λ_{vsub} has been used for syntactical studies while λ_{shuf} for semantical ones. Our results therefore establish bridges between these different (sub)communities.

Finally, an essential contribution of this work is the recognition of Open CBV as a simple and yet rich framework in between Closed and Strong CBV.

Road Map. Sect. 2 provides an overview of the different presentations of Open CBV. Sect. 3 proves the termination equivalences for λ_{vsub}, λ_{fire} and λ_{shuf}, enriched with quantitative information. Sect. 4 proves the quantitative termination equivalence of λ_{vsub} and λ_{vseq}, via an intermediate calculus $\lambda_{\mathsf{vsub}_k}$.

A longer version of this paper is available on Arxiv [3]. It contains two Appendices, one with a glossary of rewriting theory and one with omitted proofs.

Terms	$t, u, s, r ::= v \mid tu$
Values	$v, v' ::= x \mid \lambda x.t$
Evaluation Contexts	$E ::= \langle\cdot\rangle \mid tE \mid Et$

Rule at Top Level	Contextual closure
$(\lambda x.t)\lambda y.u \mapsto_{\beta_\lambda} t\{x \leftarrow \lambda y.u\}$	$E\langle t\rangle \to_{\beta_\lambda} E\langle u\rangle$ if $t \mapsto_{\beta_\lambda} u$
$(\lambda x.t)y \mapsto_{\beta_y} t\{x \leftarrow y\}$	$E\langle t\rangle \to_{\beta_y} E\langle u\rangle$ if $t \mapsto_{\beta_y} u$

Reduction $\quad \to_{\beta_v} := \to_{\beta_\lambda} \cup \to_{\beta_y}$

Fig. 1. Naïve Open CBV λ_{Plot}

2 Incarnations of Open Call-by-Value

Here we recall Naïve Open CBV, noted λ_{Plot}, and introduce the four forms of Open CBV that will be compared (λ_{fire}, λ_{vsub}, λ_{shuf}, and λ_{vseq}) together with a semantic notion (*potential valuability*) reducing Open CBV to Closed CBV. In this paper terms are always possibly open. Moreover, we focus on Open CBV and avoid on purpose to study Strong CBV (we hint at how to define it, though).

Naïve Open CBV: Plotkin's calculus λ_{Plot} [26]. Naïve Open CBV is Plotkin's weak CBV λ-calculus λ_{Plot} on possibly open terms, defined in Fig. 1. Our presentation of the rewriting is unorthodox because we split β_v-reduction into two rules, according to the kind of value (abstraction or variable). The set of terms is denoted by Λ. Terms (in Λ) are always identified up to α-equivalence and the set of the free variables of a term t is denoted by $\mathtt{fv}(t)$. We use $t\{x \leftarrow u\}$ for the term obtained by the capture-avoiding substitution of u for each free occurrence of x in t. Evaluation $\to_{\beta_v}$ is weak and non-deterministic, since in the case of an application there is no fixed order in the evaluation of the left and right subterms. As it is well-known, non-determinism is only apparent: the system is strongly confluent (see the appendix in [3] for a glossary of rewriting theory).

Proposition 1. *$\to_{\beta_y}$, $\to_{\beta_\lambda}$ and $\to_{\beta_v}$ are strongly confluent.*

Strong confluence is a remarkable property, much stronger than plain confluence. It implies that, given a term, *all* derivations to its normal form (if any) have the *same length*, and that *normalization and strong normalization coincide*, *i.e.* if there is a normalizing derivation then there are no diverging derivations. Strong confluence will also hold for λ_{fire}, λ_{vsub} and λ_{vseq}, not for λ_{shuf}.

Let us come back to the splitting of $\to_{\beta_v}$. In Closed CBV it is well-known that $\to_{\beta_y}$ is superfluous, at least as long as small-step evaluation is considered, see [5]. For Open CBV, $\to_{\beta_y}$ is instead necessary, but—as we explained in the introduction—it is not enough, which is why we shall consider extensions of λ_{Plot}. The main problem of Naïve Open CBV is that there are stuck β-redexes breaking the harmony of the system. There are three kinds of solution: those *restoring a form of harmony* (λ_{fire}), to be thought as more semantical approaches; those *removing stuck β-redexes* (λ_{vsub} and λ_{shuf}), that are more syntactical in nature; those *changing the applicative structure of terms* (λ_{vseq}), inspired by sequent calculus.

Terms and Values	As in Plotkin's Open CBV (Fig. 1)
Fireballs	$f, f', f'' ::= \lambda x.t \mid i$
Inert Terms	$i, i', i'' ::= x f_1 \dots f_n \quad n \geq 0$
Evaluation Contexts	$E ::= \langle\cdot\rangle \mid tE \mid Et$

Rule at Top Level	Contextual closure
$(\lambda x.t)(\lambda y.u) \mapsto_{\beta_\lambda} t\{x \leftarrow \lambda y.u\}$	$E\langle t\rangle \to_{\beta_\lambda} E\langle u\rangle$ if $t \mapsto_{\beta_\lambda} u$
$(\lambda x.t)i \mapsto_{\beta_i} t\{x \leftarrow i\}$	$E\langle t\rangle \to_{\beta_i} E\langle u\rangle$ if $t \mapsto_{\beta_i} u$

Reduction $\quad \to_{\beta_f} := \to_{\beta_\lambda} \cup \to_{\beta_i}$

Fig. 2. The Fireball Calculus λ_{fire}

2.1 Open Call-by-Value 1: The Fireball Calculus λ_{fire}

The Fireball Calculus λ_{fire}, defined in Fig. 2, was introduced without a name by Paolini and Ronchi Della Rocca in [25] and [28, Def. 3.1.4, p. 36] where its basic properties are also proved. We give here a presentation inspired by Accattoli and Sacerdoti Coen's [1], departing from it only for inessential, cosmetic details. Terms, values and evaluation contexts are the same as in λ_{Plot}.

The idea is to restore harmony by generalizing $\to_{\beta_y}$ to fire when the argument is a more general *inert term*—the new rule is noted $\to_{\beta_i}$. The generalization of values as to include inert terms is called *fireballs*. Actually fireballs and inert terms are defined by mutual induction (in Fig. 2). For instance, $\lambda x.y$ is a fireball as an abstraction, while x, $y(\lambda x.x)$, xy, and $(z(\lambda x.x))(zz)(\lambda y.(zy))$ are fireballs as inert terms. Note that ii' is an inert term for all inert terms i and i'. Inert terms can be equivalently defined as $i ::= x \mid if$. The main feature of an inert term is that it is open, normal and that when plugged in a context it cannot create a redex, hence the name (it is not a so-called *neutral term* because it might have redexes under abstractions). In Grégoire and Leroy's presentation [13], inert terms are called *accumulators* and fireballs are simply called values.

Evaluation is given by the fireball rule $\to_{\beta_f}$, that is the union of $\to_{\beta_\lambda}$ and $\to_{\beta_i}$. For instance, consider $t := ((\lambda y.\delta)(zz))\delta$ and $u := \delta((\lambda y.\delta)(zz))$ as in Eq. (1), p. 2: t and u are β_v-normal but they diverge when evaluated in λ_{fire}, as desired: $t \to_{\beta_i} \delta\delta \to_{\beta_\lambda} \delta\delta \to_{\beta_\lambda} \dots$ and $u \to_{\beta_i} \delta\delta \to_{\beta_\lambda} \delta\delta \to_{\beta_\lambda} \dots$.

The distinguished, key property of λ_{fire} is (for any $t \in \Lambda$):

Proposition 2 (Open Harmony). *t is β_f-normal iff t is a fireball.*

The advantage of λ_{fire} is its simple notion of normal form, *i.e.* fireballs, that have a clean syntactic description akin to that for call-by-name. The other calculi will lack a nice, natural notion of normal form. The drawback of the fireball calculus—and probably the reason why its importance did not emerge before—is the fact that as a strong calculus it is not confluent: this is due to the fact that fireballs are not closed by substitution (see [28, p. 37]). Indeed, if evaluation is strong, the following critical pair cannot be joined, where $t := (\lambda y.I)(\delta\delta)$ and $I := \lambda z.z$ is the identity combinator:

$$I \,{}_{\beta_\lambda}\!\leftarrow (\lambda x.I)\delta \,{}_{\beta_i}\!\leftarrow (\lambda x.(\lambda y.I)(xx))\delta \to_{\beta_\lambda} t \to_{\beta_\lambda} t \to_{\beta_\lambda} \dots \tag{2}$$

On the other hand, as long as evaluation is weak (that is the case we consider) everything works fine—the strong case can then be caught by repeatedly iterating the weak one under abstraction, once a weak normal form has been obtained (thus forbidding the left part of (2)). In fact, the weak evaluation of λ_{fire} has a simple rewriting theory, as next proposition shows. In particular it is strongly confluent.

Proposition 3 (Basic Properties of λ_{fire}).

1. *$\to_{\beta_i}$ is strongly normalizing and strongly confluent.*
2. *$\to_{\beta_\lambda}$ and $\to_{\beta_i}$ strongly commute.*
3. *$\to_{\beta_f}$ is strongly confluent, and all β_f-normalizing derivations d from $t \in \Lambda$ (if any) have the same length $|d|_{\beta_f}$, the same number $|d|_{\beta_\lambda}$ of β_λ-steps, and the same number $|d|_{\beta_i}$ of β_i-steps.*

2.2 Open Call-by-Value 2: The Value Substitution Calculus λ_{vsub}

Rewriting Preamble: Creations of Type 1 and 4. The problem with stuck β-redexes can be easily understood at the rewriting level as an issue about creations. According to Lévy [19], in the ordinary CBN λ-calculus redexes can be created in 3 ways. Creations of type 1 take the following form

$$((\lambda x.\lambda y.t)r)s \to_\beta (\lambda y.t\{x\leftarrow r\})s$$

where the redex involving λy and s has been created by the β-step. In Naïve Open CBV if r is a normal form and not a value then the creation cannot take place, blocking evaluation. This is the problem concerning the term t in Eq. (1), p. 2. In CBV there is another form of creation—of *type 4*—not considered by Lévy:

$$(\lambda x.t)((\lambda y.v)v') \to_{\beta_v} (\lambda x.t)(v\{y\leftarrow v'\})$$

i.e. a reduction in the argument turns the argument itself into a value, creating a β_v-redex. As before, in an open setting v' may be replaced by a normal form that is not a value, blocking the creation of type 4. This is exactly the problem concerning the term u in Eq. (1), p. 2.

The proposals of this and the next sections introduce some way to enable creations of type 1 and 4, without substituting stuck β-redexes nor inert terms.

The *value substitution calculus* λ_{vsub} of Accattoli and Paolini [4,2] was introduced as a calculus for Strong CBV inspired by linear logic proof nets. In Fig. 3 we present its adaptation to Open CBV, obtained by simply removing abstractions from evaluation contexts. It extends the syntax of terms with the constructor $[x\leftarrow u]$, called *explicit substitution* (shortened ES, to not be confused with the meta-level substitution $\{x\leftarrow u\}$). A vsub-term $t[x\leftarrow u]$ represents the delayed substitution of u for x in t, *i.e.* stands for $\mathtt{let}\ x = u\ \mathtt{in}\ t$. So, $t[x\leftarrow u]$ binds the free occurrences of x in t. The set of vsub-terms—identified up to α-equivalence—is denoted by Λ_{vsub} (clearly $\Lambda \subsetneq \Lambda_{\mathsf{vsub}}$).

ES are used to remove stuck β-redexes: the idea is that β-redexes can be fired whenever—even if the argument is not a (vsub-)value—by means of the

$$
\begin{array}{rc}
\mathsf{vsub}\text{-Terms} & t,u,s ::= v \mid tu \mid t[x{\leftarrow}u] \\
\mathsf{vsub}\text{-Values} & v ::= x \mid \lambda x.t \\
\text{Evaluation Contexts} & E ::= \langle\cdot\rangle \mid tE \mid Et \mid E[x{\leftarrow}u] \mid t[x{\leftarrow}E] \\
\text{Substitution Contexts} & L ::= \langle\cdot\rangle \mid L[x{\leftarrow}u]
\end{array}
$$

$$
\begin{array}{cc}
\textsc{Rule at Top Level} & \textsc{Contextual closure} \\
L\langle\lambda x.t\rangle u \mapsto_{\mathsf{m}} L\langle t[x{\leftarrow}u]\rangle & E\langle t\rangle \to_{\mathsf{m}} E\langle u\rangle \quad \text{if } t \mapsto_{\mathsf{m}} u \\
t[x{\leftarrow}L\langle\lambda y.u\rangle] \mapsto_{\mathsf{e}_\lambda} L\langle t\{x{\leftarrow}\lambda y.u\}\rangle & E\langle t\rangle \to_{\mathsf{e}_\lambda} E\langle u\rangle \quad \text{if } t \mapsto_{\mathsf{e}_\lambda} u \\
t[x{\leftarrow}L\langle y\rangle] \mapsto_{\mathsf{e}_y} L\langle t\{x{\leftarrow}y\}\rangle & E\langle t\rangle \to_{\mathsf{e}_y} E\langle u\rangle \quad \text{if } t \mapsto_{\mathsf{e}_y} u
\end{array}
$$

$$
\text{Reductions} \qquad \to_{\mathsf{e}} := \to_{\mathsf{e}_\lambda} \cup \to_{\mathsf{e}_y}, \quad \to_{\mathsf{vsub}} := \to_{\mathsf{m}} \cup \to_{\mathsf{e}}
$$

Fig. 3. The Value Substitution Calculus λ_{vsub}

multiplicative rule $\to_{\mathsf{m}}$; however the argument is not substituted but placed in a ES. The actual substitution is done only when the content of the ES is a vsub-value, by means of the *exponential rule* $\to_{\mathsf{e}}$. These two rules are sometimes noted $\to_{\mathsf{dB}}$ (β at a distance) and $\to_{\mathsf{vs}}$ (substitution by value)—the names we use here are due to the interpretation of the calculus into linear logic proof-nets, see [2]. A characteristic feature coming from such an interpretation is that the rewriting rules are contextual, or *at a distance*: they are generalized as to act up to a list of substitutions (noted L, from *L*ist). Essentially, stuck β-redexes are turned into ES and then ignored by the rewriting rules—this is how creations of type 1 and 4 are enabled. For instance, the terms $t := ((\lambda y.\delta)(zz))\delta$ and $u := \delta((\lambda y.\delta)(zz))$ (as in Eq. (1), p. 2) are e-normal but $t \to_{\mathsf{m}} \delta[y{\leftarrow}zz]\delta \to_{\mathsf{m}} (xx)[x{\leftarrow}\delta][y{\leftarrow}zz] \to_{\mathsf{e}} (\delta\delta)[y{\leftarrow}zz] \to_{\mathsf{m}} (xx)[x{\leftarrow}\delta][y{\leftarrow}zz] \to_{\mathsf{e}} (\delta\delta)[y{\leftarrow}zz] \to_{\mathsf{m}} \dots$ and similarly for u.

The drawback of λ_{vsub} is that it requires explicit substitutions. The advantage of λ_{vsub} is its simple and well-behaved rewriting theory, even simpler than the rewriting for λ_{fire}, since every rule terminates separately (while β_λ does not)—in particular strong confluence holds. Moreover, the theory has a sort of flexible second level given by a notion of structural equivalence, coming up next.

Proposition 4 (Basic Properties of λ_{vsub}, [4]).

1. $\to_{\mathsf{m}}$ *and* $\to_{\mathsf{e}}$ *are strongly normalizing and strongly confluent (separately).*
2. $\to_{\mathsf{m}}$ *and* $\to_{\mathsf{e}}$ *strongly commute.*
3. $\to_{\mathsf{vsub}}$ *is strongly confluent, and all* vsub*-normalizing derivations* d *from* $t \in \Lambda_{\mathsf{vsub}}$ *(if any) have the same length* $|d|_{\mathsf{vsub}}$*, the same number* $|d|_{\mathsf{e}}$ *of* e*-steps, and the same number* $|d|_{\mathsf{m}}$ *of* m*-steps*
4. *Let* $t \in \Lambda$*. For any* vsub*-derivation* d *from* t*,* $|d|_{\mathsf{e}} \leq |d|_{\mathsf{m}}$*.*

Structural Equivalence. The theory of λ_{vsub} comes with a notion of structural equivalence $\equiv$, that equates vsub-terms that differ only for the position of ES. The basic idea is that the action of an ES via the exponential rule depends on the position of the ES itself only for inessential details (as long as the scope of binders is respected), namely the position of other ES, and thus can be abstracted away. A strong justification for the equivalence comes from the linear logic interpretation of λ_{vsub}, in which structurally equivalent vsub-terms translate to the same (recursively typed) proof net, see [2].

Structural equivalence $\equiv$ is defined as the least equivalence relation on Λ_{vsub} closed by evaluation contexts (see Fig. 3) and generated by the following axioms:

$$\begin{array}{rll} t[y{\leftarrow}s][x{\leftarrow}u] \equiv_{\mathrm{com}} t[x{\leftarrow}u][y{\leftarrow}s] & & \text{if } y \notin \mathtt{fv}(u) \text{ and } x \notin \mathtt{fv}(s) \\ t\,s[x{\leftarrow}u] \equiv_{@\mathrm{r}} (ts)[x{\leftarrow}u] & & \text{if } x \notin \mathtt{fv}(t) \\ t[x{\leftarrow}u]s \equiv_{@\mathrm{l}} (ts)[x{\leftarrow}u] & & \text{if } x \notin \mathtt{fv}(s) \\ t[x{\leftarrow}u[y{\leftarrow}s]] \equiv_{[\cdot]} t[x{\leftarrow}u][y{\leftarrow}s] & & \text{if } y \notin \mathtt{fv}(t) \end{array}$$

We set $\to_{\mathsf{vsub}_\equiv} := \equiv\!\!\to_{\mathsf{vsub}}\!\!\equiv$ (*i.e.* for all $t, r \in \Lambda_{\mathsf{vsub}}$: $t \to_{\mathsf{vsub}_\equiv} r$ iff $t \equiv u \to_{\mathsf{vsub}} s \equiv r$ for some $u, s \in \Lambda_{\mathsf{vsub}}$). The notation $\to^+_{\mathsf{vsub}_\equiv}$ keeps its usual meaning, while $\to^*_{\mathsf{vsub}_\equiv}$ stands for $\equiv \cup \to^+_{\mathsf{vsub}_\equiv}$, *i.e.* a $\mathsf{vsub}_\equiv$-derivation of length zero can apply $\equiv$ and is not just the identity. As $\equiv$ is reflexive, $\to_{\mathsf{vsub}} \subsetneq \to_{\mathsf{vsub}_\equiv}$.

The rewriting theory of λ_{vsub} enriched with structural equivalence $\equiv$ is remarkably simple, as next lemma shows. In fact, $\equiv$ commutes with evaluation, and can thus be postponed. Additionally, the commutation is *strong*, as it preserves the number and kind of steps—one says that it is a *strong bisimulation* (with respect to $\to_{\mathsf{vsub}}$). In particular, the equivalence is not needed to compute and it does not break, or make more complex, any property of λ_{vsub}. On the contrary, it enhances the flexibility of the system: it will be essential to establish simple and clean relationships with the other calculi for Open CBV.

Lemma 5 (Basic Properties of Structural Equivalence $\equiv$, [4]). *Let $t, u \in \Lambda_{\mathsf{vsub}}$ and $\mathsf{x} \in \{\mathsf{m}, \mathsf{e}_\lambda, \mathsf{e}_y, \mathsf{e}, \mathsf{vsub}\}$.*

1. Strong Bisimulation of $\equiv$ wrt $\to_{\mathsf{vsub}}$*: if $t \equiv u$ and $t \to_{\mathsf{x}} t'$ then there exists $u' \in \Lambda_{\mathsf{vsub}}$ such that $u \to_{\mathsf{x}} u'$ and $t' \equiv u'$.*
2. Postponement of $\equiv$ wrt $\to_{\mathsf{vsub}}$*: if $d\colon t \to^*_{\mathsf{vsub}_\equiv} u$ then there are $s \equiv u$ and $e\colon t \to^*_{\mathsf{vsub}} s$ such that $|d| = |e|$, $|d|_{\mathsf{e}_\lambda} = |e|_{\mathsf{e}_\lambda}$, $|d|_{\mathsf{e}_y} = |e|_{\mathsf{e}_y}$ and $|d|_{\mathsf{m}} = |e|_{\mathsf{m}}$.*
3. Normal Forms*: if $t \equiv u$ then t is x-normal iff u is x-normal.*
4. Strong confluence*: $\to_{\mathsf{vsub}_\equiv}$ is strongly confluent.*

2.3 Open Call-by-Value 3: The Shuffling Calculus λ_{shuf}

The calculus introduced by Carraro and Guerrieri in [8], and here deemed *Shuffling Calculus*, has the same syntax of terms as Plotkin's calculus. Two additional commutation rules help $\to_{\beta_v}$ to deal with stuck β-redexes, by shuffling constructors so as to enable creations of type 1 and 4. As for λ_{vsub}, λ_{shuf} was actually introduced, and then used in [8,14,15], to study Strong CBV. In Fig. 4 we present its adaptation to Open CBV, based on *balanced contexts*, a special notion of evaluation contexts. The reductions $\to_{\sigma^\flat}$ and $\to_{\beta^\flat_v}$ are non-deterministic and—because of balanced contexts—can reduce under abstractions, but they are *morally* weak: they reduce under a λ only when the λ is applied to an argument. Note that the condition $x \notin \mathtt{fv}(s)$ (resp. $x \notin \mathtt{fv}(v)$) in the definition of the shuffling rule $\mapsto_{\sigma_1}$ (resp. $\mapsto_{\sigma_3}$) can always be fulfilled by α-conversion.

The rewriting (shuffling) rules $\to_{\sigma^\flat_1}$ and $\to_{\sigma^\flat_3}$ unblock stuck β-redexes. For instance, consider the terms $t := ((\lambda y.\delta)(zz))\delta$ and $u := \delta((\lambda y.\delta)(zz))$ where

Terms and Values	As in Plotkin's Open CBV (Fig. 1)
Balanced Contexts	$B ::= \langle\cdot\rangle \mid tB \mid Bt \mid (\lambda x.B)t$

Rule at Top Level	Contextual closure
$((\lambda x.t)u)s \mapsto_{\sigma_1} (\lambda x.ts)u,\ x \notin \mathtt{fv}(s)$	$B\langle t\rangle \to_{\sigma_1^\flat} B\langle u\rangle$ if $t \mapsto_{\sigma_1} u$
$v((\lambda x.s)u) \mapsto_{\sigma_3} (\lambda x.vs)u,\ x \notin \mathtt{fv}(v)$	$B\langle t\rangle \to_{\sigma_3^\flat} B\langle u\rangle$ if $t \mapsto_{\sigma_3} u$
$(\lambda x.t)v \mapsto_{\beta_v} t\{x \leftarrow v\}$	$B\langle t\rangle \to_{\beta_v^\flat} B\langle u\rangle$ if $t \mapsto_{\beta_v} u$

Reductions $\quad \to_{\sigma^\flat} := \to_{\sigma_1^\flat} \cup \to_{\sigma_3^\flat}, \quad \to_{\mathsf{shuf}} := \to_{\beta_v^\flat} \cup \to_{\sigma^\flat}$

Fig. 4. The Shuffling Calculus λ_{shuf}

$\delta := \lambda x.xx$ (as in Eq. (1), p. 2): t and u are $\beta_v^\flat$-normal but $t \to_{\sigma_1^\flat} (\lambda y.\delta\delta)(zz) \to_{\beta_v^\flat} (\lambda y.\delta\delta)(zz) \to_{\beta_v^\flat} \dots$ and $u \to_{\sigma_3^\flat} (\lambda y.\delta\delta)(zz) \to_{\beta_v^\flat} (\lambda x.\delta\delta)(zz) \to_{\beta_v^\flat} \dots$.

The similar shuffling rules in CBN, better known as Regnier's σ-rules [27], are *contained* in CBN β-equivalence, while in Open (and Strong) CBV they are more interesting because they are not contained into (*i.e.* they enrich) β_v-equivalence.

The advantage of λ_{shuf} is with respect to denotational investigations. In [8], λ_{shuf} is indeed used to prove various semantical results in connection to linear logic, resource calculi, and the notion of Taylor expansion due to Ehrhard. In particular, in [8] it has been proved the adequacy of λ_{shuf} with respect to the relational model induced by linear logic: a by-product of our paper is the extension of this adequacy result to all incarnations of Open CBV. The drawback of λ_{shuf} is its technical rewriting theory. We summarize some properties of λ_{shuf}:

Proposition 6 (Basic Properties of λ_{shuf}, [8]).

1. *Let $t, u, s \in \Lambda$. If $t \to_{\beta_v^\flat} u$ and $t \to_{\sigma^\flat} s$ then $u \neq s$.*
2. *$\to_{\sigma^\flat}$ is strongly normalizing and (not strongly) confluent.*
3. *$\to_{\mathsf{shuf}}$ is (not strongly) confluent.*
4. *Let $t \in \Lambda$: t is strongly* shuf*-normalizable iff t is* shuf*-normalizable.*

In contrast to λ_{fire} and λ_{vsub}, λ_{shuf} is not strongly confluent and not all shuf-normalizing derivations (if any) from a given term have the same length (consider, for instance, all shuf-normalizing derivations from $(\lambda y.z)(\delta(zz))\delta$). Nonetheless, normalization and strong normalization still coincide in λ_{shuf} (Prop. 6.4), and Cor. 18 in Sect. 3 will show that the discrepancy is encapsulated inside the additional shuffling rules, since all shuf-normalizing derivations (if any) from a given term have the same number of $\beta_v^\flat$-steps.

2.4 Open Call-by-Value 4: The Value Sequent Calculus λ_{vseq}

A more radical approach to the removal of stuck β-redexes is provided by what is here called the *Value Sequent Calculus* λ_{vseq}, defined in Fig. 5. In λ_{vseq}, it is the applicative structure of terms that is altered, by replacing the application constructor with more constructs, namely commands c and environments e. Morally, λ_{vseq} looks at a sequence of applications from the head, that is the value on the left of a command $\langle v \,|\, e\rangle$ rather than from the tail as in natural deduction.

Commands	$c, c' ::= \langle v \,|\, e \rangle$
Values	$v, v' ::= x \mid \lambda x.c$
Environments	$e, e' ::= \epsilon \mid \tilde{\mu} x.c \mid v \cdot e$
Command Evaluation Contexts	$C ::= \langle \cdot \rangle \mid D\langle \tilde{\mu} x.C \rangle$
Environment Evaluation Contexts	$D ::= \langle v \,|\, \langle \cdot \rangle \rangle \mid D\langle v \cdot \langle \cdot \rangle \rangle$

Rule at Top Level	Contextual closure
$\langle \lambda x.c \,|\, v \cdot e \rangle \mapsto_{\bar{\lambda}} \langle v \,|\, (\tilde{\mu} x.c) @ e \rangle$	$C\langle c \rangle \to_{\bar{\lambda}} C\langle c' \rangle$ if $c \mapsto_{\bar{\lambda}} c'$
$\langle v \,|\, \tilde{\mu} x.c \rangle \mapsto_{\tilde{\mu}} c\{x \leftarrow v\}$	$C\langle c \rangle \to_{\tilde{\mu}} C\langle c' \rangle$ if $c \mapsto_{\tilde{\mu}} c'$

Reduction $\quad \to_{\mathsf{vseq}} := \to_{\bar{\lambda}} \cup \to_{\tilde{\mu}}$

Fig. 5. The Value Sequent Calculus λ_{vseq}

In fact, λ_{vseq} is a handy presentation of the intuitionistic fragment of $\bar{\lambda}\tilde{\mu}$, that in turn is the CBV fragment of $\overline{\lambda}\mu\tilde{\mu}$, a calculus obtained as the computational interpretation of a sequent calculus for classical logic. Both $\bar{\lambda}\tilde{\mu}$ and $\overline{\lambda}\mu\tilde{\mu}$ are due to Curien and Herbelin [9], see [6,10] for further investigations about these systems.

A peculiar trait of the sequent calculus approach is the environment constructor $\tilde{\mu} x.c$, that is a binder for the free occurrences of x in c. It is often said that it is a sort of explicit substitution—we will see exactly in which sense, in Sect. 4.

The change of the intuitionistic variant λ_{vseq} with respect to $\bar{\lambda}\tilde{\mu}$ is that λ_{vseq} does not need the syntactic category of co-variables α, as there can be only one of them, denoted here by ϵ. From a logical viewpoint, this is due to the fact that in intuitionistic sequent calculus the right-hand-side of $\vdash$ has exactly one formula, that is neither contraction nor weakening are allowed on the right. Consequently, the binary abstraction $\lambda(x, \alpha).c$ of $\bar{\lambda}\tilde{\mu}$ is replaced by a more traditional unary one $\lambda x.c$, and substitution on co-variables is replaced by a notion of *appending of environments*, defined by mutual induction on commands and environments as follows:

$$\langle v \,|\, e' \rangle @ e := \langle v \,|\, e' @ e \rangle \qquad \epsilon @ e := e$$
$$(v \cdot e') @ e := v \cdot (e' @ e) \qquad (\tilde{\mu} x.c) @ e := \tilde{\mu} y.(c\{x \leftarrow y\} @ e) \text{ with } y \notin \mathtt{fv}(c) \cup \mathtt{fv}(e)$$

Essentially, $c @ e$ is a capture-avoiding substitution of e for the only occurrence of ϵ in c that is out of all abstractions, standing for the output of the term. The append operation is used in the rewrite rule $\to_{\bar{\lambda}}$ of λ_{vseq} (Fig. 5). Strong CBV can be obtained by simply extending the grammar of evaluation contexts to commands under abstractions.

We will provide a translation from λ_{vsub} to λ_{vseq} that, beyond termination equivalence, will show that switching to a sequent calculus representation is equivalent to a transformation in administrative normal form [29].

The advantage of λ_{vseq} is that it avoids both rules at a distance and shuffling rules. The drawback of λ_{vseq} is that, syntactically, it requires to step out of the λ-calculus. We will show in Sect. 4 how to reformulate it as a fragment of λ_{vsub}, *i.e.* in natural deduction. However, it will still be necessary to restrict the application constructor, thus preventing the natural way of writing terms.

The rewriting of λ_{vseq} is very well-behaved, in particular it is strongly confluent and every rewriting rule terminates separately.

Proposition 7 (Basic properties of λ_{vseq}).

1. *$\to_{\bar{\lambda}}$ and $\to_{\tilde{\mu}}$ are strongly normalizing and strongly confluent (separately).*
2. *$\to_{\bar{\lambda}}$ and $\to_{\tilde{\mu}}$ strongly commute.*
3. *$\to_{\mathsf{vseq}}$ is strongly confluent, and all* vseq*-normalizing derivations d from a command c (if any) have the same length $|d|$, the same number $|d|_{\tilde{\mu}}$ of $\tilde{\mu}$-steps, and the same number $|d|_{\bar{\lambda}}$ of $\bar{\lambda}$-steps.*

2.5 Variations on a Theme

Reducing Open to Closed Call-by-Value: Potential Valuability. Potential valuability relates Naïve Open CBV to Closed CBV via a meta-level substitution closing open terms: a (possibly open) term t is *potentially valuable* if there is a substitution of (closed) *values* for its free variables, for which it β_v-evaluates to a (closed) *value*.[3] In Naïve Open CBV, potentially valuable terms do not coincide with normalizable terms because of premature β_v-normal forms—such as t and u in Eq. (1) at p. 2— which are not potentially valuable.

Paolini, Ronchi Della Rocca and, later, Pimentel [25,23,28,24,22] gave several operational, logical, and semantical characterizations of potentially valuable terms in Naïve Open CBV. In particular, in [25,28] it is proved that a term is potentially valuable in Plotkin's Naïve Open CBV iff its normalizable in λ_{fire}.

Potentially valuable terms can be defined for every incarnation of Open CBV: it is enough to update the notions of evaluation and values in the above definition to the considered calculus. This has been done for λ_{shuf} in [8], and for λ_{vsub} in [4]. For both calculi it has been proved that, in the weak setting, potentially valuable terms coincides with normalizable terms. In [15], it has been proved that Plotkin's potentially valuable terms coincide with shuf-potentially valuable terms (which coincide in turn with shuf-normalizable terms). Our paper makes a further step: proving that termination coincides for λ_{fire}, λ_{vsub}, λ_{shuf}, and λ_{vseq} it implies that all their notions of potential valuability coincide with Plotkin's, *i.e.* there is just one notion of potential valuability for Open (and Strong) CBV.

Open CBV 5, 6, 7, ... The literature contains many other calculi for CBV, usually presented for Strong CBV and easily adaptable to Open CBV. Some of them have `let`-expressions (avatars of ES) and all of them have rules permuting constructors, therefore they lie somewhere in between λ_{vsub} and λ_{shuf}. Often, they have been developed for other purposes, usually to investigate the relationship with monad or CPS translations. Moggi's equational theory [21] is a classic standard of reference, known to coincide with that of Sabry and Felleisen [29], Sabry and Wadler [30], Dychoff and Lengrand [12], Herbelin and Zimmerman [16] and Maraist et al's λ_{let} in [20]. In [4], λ_{vsub} modulo $\equiv$ is shown to be termination equivalent to Herbelin and Zimmerman's calculus, and to strictly contain its equational theory, and thus Moggi's. At the level of rewriting these presentations of Open CBV are all more involved than those that we consider here. Their equivalence to our calculi can be shown along the lines of that of λ_{shuf} with λ_{vsub}.

[3] Potential valuability for Plotkin's CBV λ-calculus can be equivalently defined using weak or strong β_v-reduction: it is the same notion for Naïve Open and Strong CBV.

3 Quantitative Equivalence of λ_{fire}, λ_{vsub}, and λ_{shuf}

Here we show the equivalence with respect to termination of λ_{fire}, λ_{vsub}, and λ_{shuf}, enriched with quantitative information on the number of steps.

On the Proof Technique. We show that termination in λ_{vsub} implies termination in λ_{fire} and λ_{shuf} by studying simulations of λ_{fire} and λ_{shuf} into λ_{vsub}. To prove the converse implications we do not use inverse simulations. Alternatively, we show that β_f- and shuf-normal forms are essentially projected into vsub-normal forms, so that if evaluation terminates in λ_{fire} or λ_{shuf} then it also terminates on λ_{vsub}.

Such a simple technique works because in the systems under study *normalization and strong normalization coincide*: if there is a normalizing derivation from a given term t then there are no diverging derivations from t (for λ_{vsub} and λ_{fire} it follows from strong confluence, for λ_{shuf} is given by Prop. 6.4). This fact is also the reason why the statements of our equivalences (forthcoming Cor. 13 and Cor. 17) address a single derivation from t rather than considering *all* derivations from t. Moreover, for any calculus, all normalizing derivations from t have the same number of steps (in λ_{shuf} it holds for $\beta_v^{\flat}$-steps, see Cor. 18), hence also the quantitative claims of Cor. 13 and Cor. 17 hold actually for *all* normalizing derivations from t.

In both simulations, the structural equivalence $\equiv$ of λ_{vsub} plays a role.

3.1 Equivalence of λ_{fire} and λ_{vsub}

A single β_v-step $(\lambda x.t)v \to_{\beta_v} t\{x\leftarrow v\}$ is simulated in λ_{vsub} by two steps: $(\lambda x.t)v \to_{\mathsf{m}} t[x\leftarrow v] \to_{\mathsf{e}} t\{x\leftarrow v\}$, *i.e.* a m-step that creates a ES, and a e-step that turns the ES into the meta-level substitution performed by the β_v-step. The simulation of an inert step of λ_{fire} is instead trickier, because in λ_{vsub} there is no rule to substitute an inert term, if it is not a variable. The idea is that an inert step $(\lambda x.t)i \to_{\beta_i} t\{x\leftarrow i\}$ is simulated only by $(\lambda x.t)i \to_{\mathsf{m}} t[x\leftarrow i]$, *i.e.* only by the m-step that creates the ES, and such a ES will never be fired—so the simulation is up to the unfolding of substitutions containing inert terms (defined right next). Everything works because of the key property of inert terms: they are normal and their substitution cannot create redexes, so it is useless to substitute them.

The *unfolding* of a vsub-term t is the term $t{\downarrow}$ obtained from t by turning ES into meta-level substitutions; it is defined by:

$$x{\downarrow} := x \qquad (tu){\downarrow} := t{\downarrow}\, u{\downarrow} \qquad (\lambda x.t){\downarrow} := \lambda x.t{\downarrow} \qquad (t[x\leftarrow u]){\downarrow} := t{\downarrow}\{x\leftarrow u{\downarrow}\}$$

For all $t, u \in \Lambda_{\mathsf{vsub}}$, $t \equiv u$ implies $t{\downarrow} = u{\downarrow}$. Also, $t{\downarrow} = t$ iff $t \in \Lambda$.

In the simulation we are going to show, structural equivalence $\equiv$ plays a role. It is used to *clean* the vsub-terms (with ES) obtained by simulation, putting them in a canonical form where ES do not appear among other constructors.

A vsub-term is *clean* if it has the form $u[x_1\leftarrow i_1]\dots[x_n\leftarrow i_n]$ (with $n \in \mathbb{N}$), $u \in \Lambda$ is called the *body*, and $i_1, \dots, i_n \in \Lambda$ are inert terms. Clearly, any term (as it is without ES) is clean. We first show how to simulate a single fireball step.

Lemma 8 (Simulation of a β_f-Step in λ_{vsub}). *Let $t, u \in \Lambda$.*

1. *If* $t \to_{\beta_\lambda} u$ *then* $t \to_{\mathsf{m}} \to_{\mathsf{e}_\lambda} u$.
2. *If* $t \to_{\beta_i} u$ *then* $t \to_{\mathsf{m}} \equiv s$, *with* $s \in \Lambda_{\mathsf{vsub}}$ *clean and* $s{\downarrow} = u$.

We cannot simulate derivations by iterating Lemma 8, because the starting term t has no ES but the simulation of inert steps introduces ES. Hence, we have to generalize Lemma 8 up to the unfolding of ES. In general, unfolding ES is a dangerous operation with respect to (non-)termination, as it may erase a diverging subterm (*e.g.* $t := x[y \leftarrow \delta\delta]$ is vsub-divergent and $t{\downarrow} = x$ is normal). In our case, however, the simulation produces clean vsub-terms, so the unfolding is safe since it can erase only inert terms and cannot create, erase, nor carry redexes.

By means of a technical lemma (see the appendix in [3]) we obtain:

Lemma 9 (Projection of a β_f-Step on $\to_{\mathsf{vsub}}$ via Unfolding). *Let* t *be a clean* vsub*-term and* u *be a term.*

1. *If* $t{\downarrow} \to_{\beta_\lambda} u$ *then* $t \to_{\mathsf{m}} \to_{\mathsf{e}_\lambda} s$, *with* $s \in \Lambda_{\mathsf{vsub}}$ *clean and* $s{\downarrow} = u$.
2. *If* $t{\downarrow} \to_{\beta_i} u$ *then* $t \to_{\mathsf{m}} \equiv s$, *with* $s \in \Lambda_{\mathsf{vsub}}$ *clean and* $s{\downarrow} = u$.

Via Lemma 9 we can now simulate whole derivations (in forthcoming Thm. 12).

Simulation and Normal Forms. The next step towards the equivalence is to relate normal forms in λ_{fire} (aka fireballs) to those in λ_{vsub}. The relationship is not perfect, since the simulation does not directly map the former to the latter—we have to work a little bit more. First of all, let us characterize the terms in λ_{vsub} obtained by projecting normalizing derivations (that always produce a fireball).

Lemma 10. *Let* t *be a clean* vsub*-term. If* $t{\downarrow}$ *is a fireball, then* t *is* $\{\mathtt{m}, \mathsf{e}_\lambda\}$*-normal and its body is a fireball.*

Now, a $\{\mathtt{m}, \mathsf{e}_\lambda\}$-normal form t morally is vsub-normal, as $\to_{\mathsf{e}_y}$ terminates (Prop. 4.1) and it cannot create $\{\mathtt{m}, \mathsf{e}_\lambda\}$-redexes. The part about creations is better expressed as a postponement property.

Lemma 11 (Linear Postponement of $\to_{\mathsf{e}_y}$). *Let* $t, u \in \Lambda_{\mathsf{vsub}}$. *If* $d \colon t \to^*_{\mathsf{vsub}} u$ *then* $e \colon t \to^*_{\mathtt{m},\mathsf{e}_\lambda} \to^*_{\mathsf{e}_y} u$ *with* $|e|_{\mathsf{vsub}} = |d|_{\mathsf{vsub}}$, $|e|_{\mathsf{m}} = |d|_{\mathsf{m}}$, $|e|_{\mathsf{e}} = |d|_{\mathsf{e}}$ *and* $|e|_{\mathsf{e}_\lambda} \geq |d|_{\mathsf{e}_\lambda}$.

The next theorem puts all the pieces together.

Theorem 12 (Quantitative Simulation of λ_{fire} in λ_{vsub}). *Let* $t, u \in \Lambda$. *If* $d \colon t \to^*_{\beta_f} u$ *then there are* $s, r \in \Lambda_{\mathsf{vsub}}$ *and* $e \colon t \to^*_{\mathsf{vsub}} r$ *such that*

1. Qualitative Relationship*:* $r \equiv s$, $u = s{\downarrow} = r{\downarrow}$ *and* s *is clean;*
2. Quantitative Relationship*:*
   1. Multiplicative Steps: $|d|_{\beta_f} = |e|_{\mathsf{m}}$*;*
   2. Exponential (Abstraction) Steps: $|d|_{\beta_\lambda} = |e|_{\mathsf{e}_\lambda} = |e|_{\mathsf{e}}$.
3. Normal Forms*: if* u *is* β_f*-normal then there exists* $g \colon r \to^*_{\mathsf{e}_y} q$ *such that* q *is a* vsub*-normal form and* $|g|_{\mathsf{e}_y} \leq |e|_{\mathsf{m}} - |e|_{\mathsf{e}_\lambda}$.

Corollary 13 (Linear Termination Equivalence of λ_{vsub} and λ_{fire}). *Let* $t \in \Lambda$. *There is a* β_f*-normalizing derivation* d *from* t *iff there is a* vsub*-normalizing derivation* e *from* t. *Moreover,* $|d|_{\beta_f} \leq |e|_{\mathsf{vsub}} \leq 2|d|_{\beta_f}$*, i.e. they are linearly related.*

The number of β_f-steps in λ_{fire} is a reasonable cost model for Open CBV [1]. Our result implies that also *the number of* m*-steps in* λ_{vsub} *is a reasonable cost model*, since the number of m-steps is *exactly* the number of β_f-steps . This fact is quite surprising: in λ_{fire} arguments of β_f-redexes are required to be fireballs, while for m-redexes there are no restrictions on arguments, and yet in any normalizing derivation their number coincide. Note, moreover, that e-steps are linear in m-steps, but only because the initial term has no ES: in general, this is not true.

3.2 Equivalence of λ_{shuf} and λ_{vsub}

A derivation $d\colon t \to^*_{\mathsf{shuf}} u$ in λ_{shuf} is simulated via a projection on multiplicative normal forms in λ_{vsub}, *i.e.* as a derivation $\mathtt{m}(t) \to^*_{\mathsf{vsub}_\equiv} \mathtt{m}(u)$ (for any vsub-term t, its multiplicative and exponential normal forms, denoted by $\mathtt{m}(t)$ and $\mathsf{e}(t)$ respectively, exist and are unique by Prop. 4). Indeed, a $\beta_v^\flat$-step of λ_{shuf} is simulated in λ_{vsub} by a e-step followed by some m-steps to reach the m-normal form. Shuffling rules $\to_{\sigma^\flat}$ of λ_{shuf} are simulated by structural equivalence $\equiv$ in λ_{vsub}: applying $\mathtt{m}(\cdot)$ to $((\lambda x.t)u)s \to_{\sigma_1^\flat} (\lambda x.(ts))u$ we obtain exactly an instance of the axiom $\equiv_{@\mathrm{l}}$ defining $\equiv$: $\mathtt{m}(t)[x{\leftarrow}\mathtt{m}(u)]\mathtt{m}(s) \equiv_{@\mathrm{l}} (\mathtt{m}(t)\mathtt{m}(s))[x{\leftarrow}\mathtt{m}(u)]$ (with the side conditions matching exactly). Similarly, $\to_{\sigma_3^\flat}$ projects to $\equiv_{@\mathrm{r}}$ or $\equiv_{[\cdot]}$ (depending on whether v in $\to_{\sigma_3^\flat}$ is a variable or an abstraction). Therefore,

Lemma 14 (Projecting a shuf-Step on $\to_{\mathsf{vsub}_\equiv}$ via m-NF). *Let* $t, u \in \Lambda$.

1. *If* $t \to_{\sigma^\flat} u$ *then* $\mathtt{m}(t) \equiv \mathtt{m}(u)$.
2. *If* $t \to_{\beta_v^\flat} u$ *then* $\mathtt{m}(t) \to_{\mathsf{e}} \to^*_{\mathsf{m}} \mathtt{m}(u)$.

In contrast to the simulation of λ_{fire} in λ_{vsub}, here the projection of a single step can be extended to derivations without problems, obtaining that the number of $\beta_v^\flat$-steps in λ_{shuf} matches exactly the number of e-steps in λ_{vsub}. Additionally, we apply the postponement of $\equiv$ (Lemma 5.2), factoring out the use of $\equiv$ (*i.e.* of shuffling rules) without affecting the number of e-steps.

To obtain the termination equivalence we also need to study normal forms. Luckily, the case of λ_{shuf} is simpler than that of λ_{fire}, as next lemma shows.

Lemma 15 (Projection Preserves Normal Forms). *Let* $t \in \Lambda$. *If* t *is* shuf*-normal then* $\mathtt{m}(t)$ *is* vsub*-normal.*

The next theorem puts all the pieces together (for any shuf-derivation d, $|d|_{\beta_v^\flat}$ is the number of $\beta_v^\flat$-steps in d: this notion is well defined by Prop. 6.1).

Theorem 16 (Quantitative Simulation of λ_{shuf} in λ_{vsub}). *Let* $t, u \in \Lambda$. *If* $d\colon t \to^*_{\mathsf{shuf}} u$ *then there are* $s \in \Lambda_{\mathsf{vsub}}$ *and* $e\colon t \to^*_{\mathsf{vsub}} s$ *such that*

1. Qualitative Relationship*:* $s \equiv \mathtt{m}(u)$*;*
2. Quantitative Relationship (Exponential Steps)*:* $|d|_{\beta_v^\flat} = |e|_{\mathsf{e}}$*;*
3. Normal Form*: if* u *is* shuf*-normal then* s *and* $\mathtt{m}(u)$ *are* vsub*-normal.*

Corollary 17 (Termination Equivalence of λ_{vsub} and λ_{shuf}). *Let* $t \in \Lambda$. *There is a* shuf*-normalizing derivation* d *from* t *iff there is a* vsub*-normalizing derivation* e *from* t. *Moreover,* $|d|_{\beta_v^\flat} = |e|_{\mathsf{e}}$.

The obtained quantitative equivalence has an interesting corollary that shows some light on why λ_{shuf} is not strongly confluent. Our simulation maps $\beta_v^\flat$-steps in λ_{shuf} to exponential steps in λ_{vsub}, that are strongly confluent, and thus in equal number in all normalizing derivations (if any) from a given term. Therefore,

Corollary 18 (Number of $\beta_v^\flat$-Steps is Invariant). *All* shuf*-normalizing derivations from $t \in \Lambda$ (if any) have the same number of $\beta_v^\flat$-steps.*

Said differently, in λ_{shuf} normalizing derivations may have different lengths but the difference is encapsulated inside the shuffling rules $\to_{\sigma_1^\flat}$ and $\to_{\sigma_3^\flat}$.

Concerning the cost model, things are subtler for λ_{shuf}. Note that the relationship between λ_{shuf} and λ_{vsub} uses the number of e-steps, while the cost model (inherited from λ_{fire}) is the number of m-steps. Do e-steps provide a reasonable cost model? Probably not, because there is a family of terms that evaluate in exponentially more m-steps than e-steps. Details are left to a longer version.

4 Quantitative Equivalence of λ_{vsub} and λ_{vseq}, via $\lambda_{\mathsf{vsub}_k}$

The quantitative termination equivalence of λ_{vsub} and λ_{vseq} is shown in two steps: first, we identify a sub-calculus $\lambda_{\mathsf{vsub}_k}$ of λ_{vsub} equivalent to the whole of λ_{vsub}, and then show that $\lambda_{\mathsf{vsub}_k}$ and λ_{vseq} are equivalent (actually isomorphic). Both steps reuse the technique of Sect. 3, *i.e.* simulation plus study of normal forms.

4.1 Equivalence of $\lambda_{\mathsf{vsub}_k}$ and λ_{vsub}

The kernel $\lambda_{\mathsf{vsub}_k}$ of λ_{vsub} is the sublanguage of λ_{vsub} obtained by replacing the application constructor tu with the restricted form tv where the right subterm can only be a value v—*i.e.*, $\lambda_{\mathsf{vsub}_k}$ is the language of so-called *administrative normal forms* [29] of λ_{vsub}. The rewriting rules are the same of λ_{vsub}. It is easy to see that $\lambda_{\mathsf{vsub}_k}$ is stable by vsub-reduction. For lack of space, more details about $\lambda_{\mathsf{vsub}_k}$ are in the appendix of [3].

The translation $(\cdot)^+$ of λ_{vsub} into $\lambda_{\mathsf{vsub}_k}$, which simply places the argument of an application into an ES, is defined by (note that $\mathtt{fv}(t) = \mathtt{fv}(t^+)$ for all $t \in \Lambda_{\mathsf{vsub}}$):

$$\begin{array}{rlrl} x^+ &:= x & (tu)^+ &:= (t^+x)[x\leftarrow u^+] \quad \text{where } x \text{ is fresh} \\ (\lambda x.t)^+ &:= \lambda x.t^+ & t[x\leftarrow u]^+ &:= t^+[x\leftarrow u^+] \end{array}$$

Lemma 19 (Simulation). *Let $t, u \in \Lambda_{\mathsf{vsub}}$.*

1. Multiplicative*: if $t \to_{\mathsf{m}} u$ then $t^+ \to_{\mathsf{m}} \to_{\mathsf{e}_y} \equiv u^+$;*
2. Exponential*: if $t \to_{\mathsf{e}_\lambda} u$ then $t^+ \to_{\mathsf{e}_\lambda} u^+$, and if $t \to_{\mathsf{e}_y} u$ then $t^+ \to_{\mathsf{e}_y} u^+$.*
3. Structural Equivalence*: $t \equiv u$ implies $t^+ \equiv u^+$.*

The translation of a vsub-normal form is not vsub_k-normal (*e.g.* $(xy)^+ = (xz)[z\leftarrow y]$) but a further exponential normalization provides a vsub_k-normal form.

Theorem 20 (Quantitative Simulation of λ_{vsub} in $\lambda_{\mathsf{vsub}_k}$). *Let $t, u \in \Lambda_{\mathsf{vsub}}$. If $d\colon t \to_{\mathsf{vsub}}^* u$ then there are $s \in \Lambda_{\mathsf{vsub}_k}$ and $e\colon t^+ \to_{\mathsf{vsub}_k}^* s$ such that*

1. Qualitative Relationship: $s \equiv u^{+}$;
2. Quantitative Relationship:
   1. Multiplicative Steps: $|e|_{\mathsf{m}} = |d|_{\mathsf{m}}$;
   2. Exponential Steps: $|e|_{\mathsf{e}_\lambda} = |d|_{\mathsf{e}_\lambda}$ *and* $|e|_{\mathsf{e}_y} = |d|_{\mathsf{e}_y} + |d|_{\mathsf{m}}$;
3. Normal Form: *if* u *is* vsub*-normal then* s *is* m*-normal and* $\mathsf{e}(s)$ *is* vsub_k*-normal.*

Unfortunately, the length of the exponential normalization in Thm. 20.3 cannot be easily bounded, forbidding a precise quantitative equivalence. Note however that turning from λ_{vsub} to its kernel $\lambda_{\mathsf{vsub}_k}$ does not change the number of multiplicative steps: the transformation preserves the cost model.

Corollary 21 (Termination and Cost Equivalence of λ_{vsub} and $\lambda_{\mathsf{vsub}_k}$). *Let* $t \in \Lambda_{\mathsf{vsub}}$. *There exists a* vsub*-normalizing derivation* d *from* t *iff there exists a* vsub_k*-normalizing derivation* e *from* t^{+}. *Moreover,* $|d|_{\mathsf{m}} = |e|_{\mathsf{m}}$.

4.2 Equivalence of $\boldsymbol{\lambda_{\mathsf{vsub}_k}}$ and $\boldsymbol{\lambda_{\mathsf{vseq}}}$

The translation $\underline{\cdot}$ of $\lambda_{\mathsf{vsub}_k}$ into λ_{vseq} relies on an auxiliary translation $(\cdot)^{\bullet}$ of values and it is defined as follows:

$$\begin{array}{lll} x^{\bullet} := x & (\lambda x.t)^{\bullet} := \lambda x.\underline{t} & \\ \underline{v} := \langle v \,|\, \epsilon \rangle & \underline{tv} := \underline{t}@(v^{\bullet}\cdot\epsilon) & \underline{t[x\leftarrow u]} := \underline{u}@\tilde{\mu}x.\underline{t} \end{array}$$

Note the subtle mapping of ES to $\tilde{\mu}$: ES correspond to appendings of $\tilde{\mu}$ to the output of the term u to be substituted, and not of the term t where to substitute.

It is not hard to see that $\lambda_{\mathsf{vsub}_k}$ and λ_{vseq} are actually isomorphic, where the converse translation $(\cdot)^{\diamond}$, that maps values and commands to terms, and environments to evaluation contexts, is given by:

$$\begin{array}{lll} x^{\diamond} := x & \epsilon^{\diamond} := \langle\cdot\rangle & \langle v \,|\, e\rangle^{\diamond} := e^{\diamond}\langle v^{\diamond}\rangle \\ (\lambda x.c)^{\diamond} := \lambda x.c^{\diamond} & (v\cdot e)^{\diamond} := e^{\diamond}\langle\langle\cdot\rangle v^{\diamond}\rangle & (\tilde{\mu}x.c)^{\diamond} := c^{\diamond}[x\leftarrow\langle\cdot\rangle] \end{array}$$

For the sake of uniformity, we follow the same structure of the other weaker equivalences (*i.e.* simulation plus mapping of normal forms, here working smoothly) rather than proving the isomorphism formally. The simulation maps multiplicative steps to $\bar{\lambda}$ steps, whose number, then, is a reasonable cost model for λ_{vseq}.

Lemma 22 (Simulation of $\to_{\mathsf{vsub}_k}$ by $\to_{\mathsf{vseq}}$). *Let* t *and* u *be* vsub_k*-terms.*

1. Multiplicative: *if* $t \to_{\mathsf{m}} u$ *then* $\underline{t} \to_{\bar{\lambda}} \underline{u}$.
2. Exponential: *if* $t \to_{\mathsf{e}} u$ *then* $\underline{t} \to_{\tilde{\mu}} \underline{u}$.

Theorem 23 (Quantitative Simulation of $\lambda_{\mathsf{vsub}_k}$ in λ_{vseq}). *Let* t *and* u *be* vsub_k*-terms. If* $d\colon t \to^{*}_{\mathsf{vsub}_k} u$ *then there is* $e\colon \underline{t} \to^{*}_{\mathsf{vseq}} \underline{u}$ *such that*

1. Multiplicative Steps: $|d|_{\mathsf{m}} = |e|_{\bar{\lambda}}$ *(the number* $\bar{\lambda}$*-steps in* e*);*
2. Exponential Steps: $|d|_{\mathsf{e}} = |e|_{\tilde{\mu}}$ *(the number* $\tilde{\mu}$*-steps in* e*), so* $|d|_{\mathsf{vsub}_k} = |e|_{\mathsf{vseq}}$;
3. Normal Form: *if* u *is* vsub_k*-normal then* $\underline{u}$ *is* vseq*-normal.*

Corollary 24 (Linear Termination Equivalence of $\lambda_{\mathsf{vsub}_k}$ and λ_{vseq}). *Let* t *be a* vsub_k*-term. There is a* vsub_k*-normalizing derivation* d *from* t *iff there is a* vseq*-normalizing derivation* e *from* $\underline{t}$. *Moreover,* $|d|_{\mathsf{vsub}_k} = |e|_{\mathsf{vseq}}$, $|d|_{\mathsf{e}} = |e|_{\tilde{\mu}}$ *and* $|d|_{\mathsf{m}} = |e|_{\bar{\lambda}}$.

Structural Equivalence for λ_{vseq}. The equivalence of λ_{vsub} and $\lambda_{\mathsf{vsub}_k}$ relies on the structural equivalence $\equiv$ of λ_{vsub}, so it is natural to wonder how does $\equiv$ look on λ_{vseq}. The structural equivalence $\eqsim$ of λ_{vseq} is defined as the closure by evaluation contexts of the following axiom

$$D\langle\tilde{\mu}x.D'\langle\tilde{\mu}y.c\rangle\rangle \eqsim_{\tilde{\mu}\tilde{\mu}} D'\langle\tilde{\mu}y.D\langle\tilde{\mu}x.c\rangle\rangle \quad \text{where } x \notin \mathtt{fv}(D') \text{ and } y \notin \mathtt{fv}(D).$$

As expected, $\eqsim$ has, with respect to λ_{vseq}, all the properties of $\equiv$ (see Lemma 5). They are formally stated in the appendix of [3], for lack of space.

5 Conclusions and Future Work

This paper proposes Open CBV as a setting halfway between Closed CBV, the simple framework used to model programming languages such as OCaml, and Strong CBV, the less simple setting underling proof assistants such as Coq. Open CBV is a good balance: its rewriting theory is simple—in particular it is strongly confluent, as the one of Closed CBV—and it can be iterated under abstractions to recover Strong CBV, which is not possible with Closed CBV.

We compared four representative calculi for Open CBV, developed with different motivations, and showed that they share the same qualitative (termination/divergence) and quantitative (number of steps) properties with respect to termination. Therefore, they can be considered as different incarnations of the same immaterial setting, justifying the slogan *Open CBV*.

The qualitative equivalences carry semantical consequences: *the adequacy of relational semantics* for the shuffling calculus proved in [8] actually gives a semantic (and type-theoretical, since the relational model can be seen as a non-idempotent intersection type system) characterization of normalizable terms for Open CBV, *i.e.* it extends to the other three calculi. Similarly, the notion of *potential valuability* for Plotkin's CBV λ-calculus, well-studied in [25,23,28,24,22] and recalled at the end of Sect. 2, becomes a robust notion characterizing the same terms in Open (and Strong) CBV.

Quantitatively, we showed that in three out of four calculi for Open CBV, namely λ_{fire}, λ_{vsub} and λ_{vseq}, evaluation takes exactly the same number of β_f-steps, m-steps and $\bar{\lambda}$-steps, respectively. Since such a number is known to be a reasonable time cost model for λ_{fire} [1], the cost model lifts to λ_{vsub} and λ_{vseq}, showing that the cost model is robust, *i.e.* incarnation-independent. For the shuffling calculus λ_{shuf} we obtain a weaker quantitative relationship that does not allow to transfer the cost model. The $\beta_v^\flat$-steps in λ_{shuf}, indeed, match e-steps in λ_{vsub}, but not m-steps. Unfortunately, the two quantities are not necessarily polynomially related, since there is a family of terms that evaluate in exponentially more m-steps than e-steps (details are left to a longer version). Consequently, λ_{shuf} is an incarnation more apt to semantical investigations rather than complexity analyses.

Future Work. This paper is just the first step towards a new, finer understanding of CBV. We plan to purse at the least the following research directions:

1. *Equational Theories.* The four incarnations are termination equivalent but their rewriting rules do not induce the same equational theory. In particular, λ_{fire} equates more than the others, and probably too much because its theory is not a congruence, *i.e.* it is not stable by context closure. The goal is to establish the relationships between the theories and understand how to smooth the calculi as to make them both *equational* and *termination* equivalent.
2. *Abstract Machines.* Accattoli and Sacerdoti Coen introduce in [1] *reasonable* abstract machines for Open CBV, that is, implementation schemas whose overhead is proven to be polynomial, and even linear. Such machines are quite complex, especially the linear one. Starting from a fine analysis of the overhead, we are currently working on a simpler approach providing cost equivalent but much simpler abstract machines.
3. *From Open CBV to Strong CBV.* We repeatedly said that Strong CBV can be seen as an iteration of Open CBV under abstractions. This is strictly true for λ_{vsub}, λ_{shuf}, and λ_{vseq}, for which the simulations studied here lift to the strong setting. On the contrary, the definition of a good strong λ_{fire} is a subtle open issue. The natural candidate, indeed, is not confluent (but enjoys uniqueness of normal forms) and normalizes more terms than the other calculi for Strong CBV. Another delicate point is the design and the analysis of abstract machines for Strong CBV, of which there are no examples in the literature (both Grégoire and Leroy's [13] and Accattoli and Sacerdoti Coen's [1] study machines for Open CBV only).
4. *Open Bisimulations.* In [18] Lassen studies open (or normal form) bisimulations for CBV. He points out that his bisimilarity is not fully abstract with respect to contextual equivalence, and his counterexamples are all based on stuck β-redexes in Naïve Open CBV. An interesting research direction is to recast his study in Open CBV and see whether full abstraction holds or not.

Acknowledgment. Work partially supported by the A*MIDEX project ANR-11-IDEX-0001-02 funded by the "Investissements d'Avenir" French Government program, managed by the French National Research Agency (ANR), and by ANR projects ANR-12-JS02-006-01 (CoQuaS) and ANR-11-IS02-0002 (Locali).

References

1. Accattoli, B., Sacerdoti Coen, C.: On the Relative Usefulness of Fireballs. In: LICS 2015. pp. 141–155 (2015)
2. Accattoli, B.: Proof nets and the call-by-value λ-calculus. Theor. Comput. Sci. 606, 2–24 (2015)
3. Accattoli, B., Guerrieri, G.: Open Call-by-Value (Extended Version). CoRR abs/1609.00322 (2016), `http://arxiv.org/abs/1609.00322`
4. Accattoli, B., Paolini, L.: Call-by-Value Solvability, revisited. In: FLOPS. pp. 4–16 (2012)
5. Accattoli, B., Sacerdoti Coen, C.: On the Value of Variables. In: WoLLIC 2014. pp. 36–50 (2014)

6. Ariola, Z.M., Bohannon, A., Sabry, A.: Sequent calculi and abstract machines. ACM Trans. Program. Lang. Syst. 31(4) (2009)
7. Blelloch, G.E., Greiner, J.: Parallelism in Sequential Functional Languages. In: FPCA. pp. 226–237 (1995)
8. Carraro, A., Guerrieri, G.: A Semantical and Operational Account of Call-by-Value Solvability. In: FOSSACS 2014. pp. 103–118 (2014)
9. Curien, P.L., Herbelin, H.: The duality of computation. In: ICFP. pp. 233–243 (2000)
10. Curien, P., Munch-Maccagnoni, G.: The Duality of Computation under Focus. In: 6th IFIP, TCS 2010. Proceedings. vol. 323, pp. 165–181. Springer (2010)
11. Dal Lago, U., Martini, S.: The weak lambda calculus as a reasonable machine. Theor. Comput. Sci. 398(1-3), 32–50 (2008)
12. Dyckhoff, R., Lengrand, S.: Call-by-Value lambda-calculus and LJQ. J. Log. Comput. 17(6), 1109–1134 (2007)
13. Grégoire, B., Leroy, X.: A compiled implementation of strong reduction. In: (ICFP '02). pp. 235–246 (2002)
14. Guerrieri, G.: Head reduction and normalization in a call-by-value lambda-calculus. In: WPTE 2015. pp. 3–17 (2015)
15. Guerrieri, G., Paolini, L., Ronchi Della Rocca, S.: Standardization of a Call-By-Value Lambda-Calculus. In: TLCA 2015. pp. 211–225 (2015)
16. Herbelin, H., Zimmermann, S.: An operational account of Call-by-Value Minimal and Classical λ-calculus in Natural Deduction form. In: TLCA. pp. 142–156 (2009)
17. Jones, N.D., Gomard, C.K., Sestoft, P.: Partial Evaluation and Automatic Program Generation. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1993)
18. Lassen, S.: Eager Normal Form Bisimulation. In: LICS 2005. pp. 345–354 (2005)
19. Lévy, J.J.: Réductions correctes et optimales dans le lambda-calcul. Thése d'Etat, Univ. Paris VII, France (1978)
20. Maraist, J., Odersky, M., Turner, D.N., Wadler, P.: Call-by-name, Call-by-value, Call-by-need and the Linear λ-Calculus. TCS 228(1-2), 175–210 (1999)
21. Moggi, E.: Computational λ-Calculus and Monads. In: LICS '89. pp. 14–23 (1989)
22. Paolini, L., Pimentel, E., Ronchi Della Rocca, S.: Strong Normalization from an unusual point of view. Theor. Comp. Science 412(20), 1903–1915 (2011)
23. Paolini, L.: Call-by-Value Separability and Computability. In: ICTCS. pp. 74–89 (2002)
24. Paolini, L., Pimentel, E., Ronchi Della Rocca, S.: Lazy strong normalization. In: ITRS '04. Electronic Notes in Theoretical Computer Science, vol. 136C, pp. 103–116 (2005)
25. Paolini, L., Ronchi Della Rocca, S.: Call-by-value Solvability. ITA 33(6), 507–534 (1999)
26. Plotkin, G.D.: Call-by-Name, Call-by-Value and the lambda-Calculus. Theor. Comput. Sci. 1(2), 125–159 (1975)
27. Regnier, L.: Une équivalence sur les lambda-termes. TCS 2(126), 281–292 (1994)
28. Ronchi Della Rocca, S., Paolini, L.: The Parametric λ-Calculus. Springer Berlin Heidelberg (2004)
29. Sabry, A., Felleisen, M.: Reasoning about Programs in Continuation-Passing Style. Lisp and Symbolic Computation 6(3-4), 289–360 (1993)
30. Sabry, A., Wadler, P.: A Reflection on Call-by-Value. ACM Trans. Program. Lang. Syst. 19(6), 916–941 (1997)
31. Sands, D., Gustavsson, J., Moran, A.: Lambda Calculi and Linear Speedups. In: The Essence of Computation, Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones. pp. 60–84 (2002)