Mirroring Call-By-Need, or Values Acting Silly

Beniamino Accattoli, Adrienne Lancelot

Mirroring Call-By-Need, or Values Acting Silly

Beniamino Accattoli
Inria & LIX, Ecole Polytechnique, UMR 7161, France

Adrienne Lancelot
Inria & LIX, Ecole Polytechnique, UMR 7161, France
Université Paris Cité, CNRS, IRIF, Paris, France

Abstract

Call-by-need evaluation for the λ-calculus can be seen as merging the best of call-by-name and call-by-value, namely the wise erasing behaviour of the former and the wise duplicating behaviour of the latter. To better understand how duplication and erasure can be combined, we design a degenerated calculus, dubbed *call-by-silly*, that is symmetric to call-by-need in that it merges the worst of call-by-name and call-by-value, namely silly duplications by-name and silly erasures by-value.

We validate the design of the call-by-silly calculus via rewriting properties and multi types. In particular, we mirror the main theorem about call-by-need – that is, its operational equivalence with call-by-name – showing that call-by-silly and call-by-value induce the same contextual equivalence. This fact shows the blindness with respect to efficiency of call-by-value contextual equivalence. We also define a call-by-silly *strategy* and measure its length via tight multi types. Lastly, we prove that the call-by-silly strategy computes evaluation sequences of maximal length in the calculus.

2012 ACM Subject Classification Theory of computation → Lambda calculus

Keywords and phrases Lambda calculus, intersection types, call-by-value, call-by-need

Digital Object Identifier 10.4230/LIPIcs.FSCD.2024.23

Related Version *Full Version*: `https://arxiv.org/abs/2402.12078` [14]

1 Introduction

Plotkin's call-by-value and Wadsworth's call-by-need (untyped) λ-calculi were introduced in the 1970s as more application-oriented variants of the ordinary call-by-name λ-calculus [53, 49]. The simpler call-by-value (shortened to CbV) calculus has found a logical foundation in formalisms related to classical logic or linear logic [28, 44, 45, 33].

The foundation of call-by-need (CbNeed) is less developed, particularly its logical interpretation. The duality related to classical logic can accommodate CbNeed, as shown by Ariola et al. [19, 17], but it does not provide an explanation for it. Within linear logic, CbNeed is understood as a sort of *affine* CbV, according to Maraist et al. [47]. Such an interpretation however is unusual, because it does not match exactly with cut-elimination in linear logic, as for call-by-name (CbN) and CbV, and it is rather connected with *affine logic*.

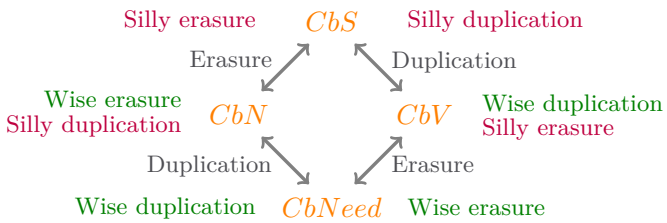
CbNeed Optimizes CbN. The main foundational theorem for CbNeed is the *operational equivalence of CbN and CbNeed* due to Ariola et al. [18], that is, the fact that they induce the same contextual equivalence, despite being based on different evaluation mechanisms. The result formalizes that CbNeed is a semantic-preserving optimization of CbN.

An elegant semantic proof of this result is given by Kesner [38]. She shows that the CbN multi type system by de Carvalho [30, 31] (considered before his seminal work also by Gardner [35] and Kfoury [43]), which characterizes termination for CbN (that is, t is typable $\Leftrightarrow$ t is CbN terminating), characterizes termination also for CbNeed. Multi types are also known as *non-idempotent intersection types*, and have strong ties with linear logic.

Duplication and Erasure, Wise and Silly. The linear logic interpretation of CbN and CbV provides some operational insights about CbNeed. By bringing to the fore duplication and erasure, linear logic underlines a second symmetry between CbN and CbV, which is independent of classical logic as it can already be observed in an intuitionistic setting. The idea is that CbN is *wise* with respect to erasure, because it never evaluates arguments that might be erased (for instance $(\lambda x.\mathtt{I})\Omega \to_{CbN} \mathtt{I}$, where $\mathtt{I}$ is the identity and Ω is the looping combinator), while it is *silly* with respect to duplications, as it repeats possibly many times the evaluation of arguments that are used at least once (for instance, $(\lambda x.xx)(\mathtt{II}) \to_{CbN} \mathtt{II}(\mathtt{II})$). Symmetrically, CbV is silly with respect to erasures, as it reduces in arguments that are not going to be used (for instance looping on $(\lambda x.\mathtt{I})\Omega$), but it is wise for duplications, as it reduces only once arguments that shall be used at least once (e.g. $(\lambda x.xx)(\mathtt{II}) \to_{CbV} (\lambda x.xx)\mathtt{I}$).

CbNeed is Pure Wiseness. In this framework, one can see CbNeed as merging the best of CbN and CbV, i.e. wise erasure and wise duplication. In [12], this insight guides Accattoli et al. in extending Kesner's above-mentioned study of CbNeed via multi types [38]. Starting from existing multi type systems for CbN and CbV, they pick out the aspects for CbN wise erasures and CbV wise duplication as to build a CbNeed multi type system. Their system characterizes CbNeed termination, as does the CbN system in [38]. Additionally, it tightly characterizes CbNeed evaluation *quantitatively*: from type derivations one can read the *exact* number of CbNeed evaluation steps, which is not possible in the CbN system.

Pure Silliness. This paper studies an unusual – and at first counter-intuitive – combination of duplication and erasure. The idea is to mix together silly erasure and silly duplication, completing the following diagram of strategies with its new *call-by-silly* (CbS) corner:



(1)

Designing the CbS calculus and a CbS evaluation strategy is of no interest for programming purposes, as CbS is desperately inefficient, by construction. It is theoretically relevant, however, because it showcases how modularly duplication and erasure can be combined. As we shall discuss, the study of CbS also contributes to the understanding of CbV.

Quantitative Goal and Micro Steps. We introduce CbS and provide evidence of its good design via contributions that mirror as much as possible the theory of CbNeed. Our ultimate test is to mirror the quantitative study of CbNeed by Accattoli et al. [12], via a new system of multi types for CbS. It is a challenging design goal because this kind of quantitative results requires a *perfect matching* between the operational and the multi types semantics.

As we explain at the end of Sect. 9, such a goal does not seem to be cleanly achievable within the λ-calculus, where duplication is *small-step*, that is, arguments are substituted on *all* the occurrences of a variable at once. It is instead attainable in a *micro-step* setting with explicit substitutions, where one copy at a time is performed. Moreover, in the λ-calculus duplication and erasure are hard to disentangle as they are both handled by β-reduction.

Our Framework. A good setting for our purpose is Accattoli and Kesner's *linear substitution calculus* (LSC) [1, 8], a calculus with explicit substitutions which is a variation over a calculus by Milner [48, 39] exploiting ideas from another work of theirs [13]. A key feature of the LSC is that it has *separate* rules for duplication and erasure, and *just one rule for each*.

The LSC comes in two dialects, CbN and CbV, designed over linear logic proof nets [3, 2]. The further CbNeed LSC is obtained by taking the duplication rule of the CbV LSC and the erasing rule of the CbN LSC. It was first defined by Accattoli et al. [5] and then studied or extended by many recent works on CbNeed [38, 21, 6, 7, 41, 23, 12, 40, 22, 15].

Our starting point is the principled definition of the new *silly (linear) substitution calculus* (SSC), the variant of the LSC obtained by mirroring the construction for CbNeed, namely by putting together the CbN rule for duplication and the CbV one for erasure. In this way, all the corners of the strategy diagram (1) fit into the same framework. Since CbNeed is usually studied with respect to weak evaluation (that is, not under abstraction), we only consider weak evaluation for the SSC (and leave the strong, unrestricted case to future work).

Contribution 1: Rewriting Properties. After defining the SSC, we prove some of the rewriting properties that are expected from every well-behaved calculus. Namely, we provide a characterization of its normal forms, that reduction is confluent, and that – as it is the case for all dialects of the LSC – the CbV erasing rule can be postponed.

Via the multi types of the next contribution, we also prove *uniform normalization* for the SSC, that is, the fact that a term is weakly normalizing (i.e. it reduces to a normal form) if and only if it is strongly normalizing (i.e. it has no diverging reductions). This is trivially true in (weak) CbV, where diverging sub-terms cannot be erased and terms with redexes cannot be duplicated, and false in CbN and CbNeed, where diverging sub-terms can be erased. In the SSC, it is true but non-trivial, because terms with redexes can be duplicated.

Contribution 2: Operational Equivalence. Next, we develop the mirrored image of the main qualitative result for CbNeed, that is, we prove the operational equivalence of CbS and CbV. We do so by mirroring Kesner's semantic proof via multi types [38].

To this purpose, we cannot use the existing multi type system for CbV, due to Ehrhard [32], as Kesner's proof is based on a system for the *slower* of the two systems, in her case CbN. Therefore, we introduce a silly multi type system, designed in a dual way to the one for CbNeed by Accattoli et al. [12].

Our system is a minor variant over a system for CbN strong normalization by Kesner and Ventura [42], and it also bears strong similarities with a system for CbV by Manzonetto et al. [46, 37]. We prove that it characterizes termination for closed terms in both CbV and CbS, from which it follows the coincidence of their respective contextual equivalences. This result shows that CbV is a semantic-preserving optimization of CbS, exactly as CbNeed is for CbN.

CbV Contextual Equivalence is Blind to Efficiency. Contextual equivalence is usually considered as *the* notion of program equivalence. This paper points out a fundamental limitation of contextual equivalence for the effect-free untyped CbV λ-calculus. In absence of effects, contextual equivalence does distinguish between CbV and CbN because of their different erasing policies (as they change the notion of termination) but it is unable to distinguish between their duplication policies (which only impact the efficiency of evaluation), because it is *blind to efficiency*: it cannot "count" how many times a sub-term is evaluated.

It is well known that, in richer frameworks with effects, CbV contextual equivalence can "count". In presence of state, indeed, more terms are discriminated, as doing once *or* twice the same operation can now change the content of a state and contexts are expressive enough to generate and distinguish these state changes [52, 25]. But it is instead not well known, or not fully digested, that in the pure setting this is not possible – thus that CbV contextual equivalence validates silly duplications – as we have repeatedly noted in discussions with surprised colleagues.

To the best of our knowledge, such a blindness to efficiency has not been properly established before. Proving contextual equivalence is technically difficult, often leading to hand-waving arguments. Moreover, figuring out a general characterization of silly duplications is tricky. We provide here an easy way out to prove CbV contextual equivalence for terms related by silly duplications: it is enough to reduce them to the same normal form in the SSC, a calculus qualitatively equivalent to CbV but based on silly duplications.

Note also that the equivalence of CbN and CbNeed is interpreted positively (despite implying that CbNeed contextual equivalence is blind to efficiency) because it gives a foundation to CbNeed. Curiously, the mirrored equivalence of CbV and CbS crystallizes instead a negative fact about pure CbV, since CbV is expected to be efficiency-sensitive.

Contribution 3: The CbS Strategy. Beyond the SSC, we specify a tricky notion of CbS *strategy*. CbS contexts are the modification of CbN contexts that also evaluate shared sub-terms when they are *no longer needed*, i.e., after all the needed copies have already been evaluated. The CbS strategy is an extension of the CbN strategy, on non-erasing steps.

We then prove that the CbS strategy is (essentially) deterministic (precisely, it is *diamond*, a weakened form of determinism), and that the CbV erasing rule of the strategy can be postponed. These usually simple results have in our case simple but lengthy proofs, because of the tricky grammar defining CbS contexts.

Contribution 4: Tight Types and (Maximal) Evaluation Lengths. Lastly, we show that the quantitative result for CbNeed by Accattoli et al. [12] is also mirrored by our silly type system. We mimic [12] and we extract the exact length of evaluations for the CbS strategy from a notion of *tight* type derivation.

We then use this quantitative result to show that the CbS strategy actually computes a *maximal* evaluation sequence in the weak SSC. This is not so surprising, given that maximal evaluations and strong normalization are related and measured with similar systems of multi types, as in Bernadet and Lengrand [24], Kesner and Ventura [42], and Accattoli et al. [9].

At the same time, however, the maximality of CbS is specific to weak evaluation and different from those results in the literature, which concern strong evaluation (that is, also under abstraction). In particular, CbS would *not* be maximal in a strong setting (which we do not treat here), as it would erase a value before evaluating it (as CbV also does). This is precisely the difference between our type system and that of Kesner and Ventura.

Related Work. The only work that might be vaguely reminiscent of ours is by Ariola et al. [19, 17], who study CbNeed with respect to the classical duality between CbN and CbV and control operators. Their work and ours however are orthogonal and incomparable: they do not obtain inefficient strategies and we do not deal with control operators.

Proofs. A long version with proofs in the Appendix is on ArXiv [14].

Language			Root rules			
Terms	t, s, u	$::= x \mid \lambda x.t \mid ts \mid t[x \leftarrow s]$	$S\langle \lambda x.t\rangle s$	$\mapsto_{\mathtt{m}}$	$S\langle t[x \leftarrow s]\rangle$	
Values	v, v'	$::= \lambda x.t$	$W\langle\!\langle x\rangle\!\rangle[x \leftarrow s]$	$\mapsto_{\mathtt{e}_W}$	$W\langle\!\langle s\rangle\!\rangle[x \leftarrow s]$	
Sub. Ctxs	S, S'	$::= \langle\cdot\rangle \mid S[x \leftarrow s]$	$t[x \leftarrow S\langle v\rangle]$	$\mapsto_{\mathtt{gcv}}$	$S\langle t\rangle$	if $x \notin \mathtt{fv}(t)$

Weak reduction $\to_{\mathtt{w}}$

Weak contexts
$W ::= \langle\cdot\rangle \mid Wt \mid tW \mid t[x \leftarrow W] \mid W[x \leftarrow s]$

$$
\begin{array}{rcl}
\to_{\mathtt{wm}} & := & W\langle \mapsto_{\mathtt{m}}\rangle \\
\to_{\mathtt{we}} & := & W\langle \mapsto_{\mathtt{e}_W}\rangle \\
\to_{\mathtt{wgcv}} & := & W\langle \mapsto_{\mathtt{gcv}}\rangle \\
\to_{\mathtt{w}} & := & \to_{\mathtt{wm}} \cup \to_{\mathtt{we}} \cup \to_{\mathtt{wgcv}}
\end{array}
$$

Figure 1 The (weak) silly (linear) substitution calculus (SSC).

2 The Weak Silly Substitution Calculus

Inception. The SSC is a variant over Accattoli and Kesner's linear substitution calculus (LSC) [1, 8], which is a micro-step λ-calculus with explicit substitutions. *Micro-step* means that substitutions act on one variable occurrence at a time, rather than *small-step*, that is, on all occurrences at the same time. The LSC exists in two main variants, CbN and CbV. The CbV variant is usually presented with small-step rules, and called *value substitution calculus*, the micro-step (or linear) variant of which appears for instance in Accattoli et al. [5, 12]. The two calculi have similar and yet different duplication and erasing rewriting rules. We refer to [5, 12] for a uniform presentation of CbN, CbV, and CbNeed in the LSC.

The SSC in Fig. 1 is obtained dually to CbNeed, taking the duplication rule $\mapsto_{\mathtt{e}_W}$ of the CbN LSC and the erasing rule $\mapsto_{\mathtt{gcv}}$ of the CbV LSC. Its evaluation is micro-step. We only define its weak evaluation, which does not enter into abstractions, as it is often done in comparative studies between CbN, CbV, and CbNeed such as [5, 12]. This sections overviews the definitions in the figure. The SSC deals with possibly open terms. The CbS strategy, to be defined in Sect. 8, shall instead deal only with closed terms.

Terms. Terms of the SSC are the same as for the LSC and extend the λ-calculus with *explicit substitutions* $t[x \leftarrow s]$ (shortened to ESs), that is a more compact notation for $\mathsf{let}\ x = s\ \mathsf{in}\ t$, but where the order of evaluation between t and s is a priori not fixed. The set $\mathtt{fv}(t)$ of *free* variables of a term t is defined as expected, in particular, $\mathtt{fv}(t[x \leftarrow s]) := (\mathtt{fv}(t) \setminus \{x\}) \cup \mathtt{fv}(s)$. Both $\lambda x.t$ and $t[x \leftarrow s]$ bind x in t, and terms are considered up to α-renaming. A term t is *closed* if $\mathtt{fv}(t) = \emptyset$, *open* otherwise. As usual, terms are identified up to α-equivalence. Meta-level capture-avoiding substitution is noted $t\{x \leftarrow s\}$. Note that values are only abstraction; this choice shall be motivated in the next section.

Contexts. Contexts are terms with exactly one occurrence of the *hole* $\langle\cdot\rangle$, an additional constant, standing for a removed sub-term. We shall use many different contexts. The most general ones in this paper are *weak contexts* W, which simply allow the hole to be anywhere but under abstraction. To define the rewriting rules, *substitution contexts* S (i.e. lists of explicit substitutions) also play a role. The main operation about contexts is *plugging* $W\langle t\rangle$ where the hole $\langle\cdot\rangle$ in context W is replaced by t. Plugging, as usual with contexts, can capture variables – for instance $((\langle\cdot\rangle t)[x \leftarrow s])\langle x\rangle = (xt)[x \leftarrow s]$. We write $W\langle\!\langle t\rangle\!\rangle$ when we want to stress that the context W does not capture the free variables of t.

Rewriting Rules. The reduction rules of the SSC are slightly unusual as they use *contexts* both to allow one to reduce redexes located in sub-terms, which is standard, *and* to define the redexes themselves, which is less standard. This approach is called *at a distance* and related to cut-elimination on proof nets (from which the terminology *multiplicative* and *exponential* to be discussed next is also taken), see Accattoli [2, 3].

The *multiplicative rule* $\mapsto_{\mathtt{m}}$ is essentially the β-rule, except that the argument goes into a new ES, rather then being immediately substituted, and that there can be a substitution context S in between the abstraction and the argument. Example: $(\lambda x.y)[y{\leftarrow}t]s \mapsto_{\mathtt{m}} y[x{\leftarrow}s][y{\leftarrow}t]$. One with on-the-fly α-renaming is $(\lambda x.y)[y{\leftarrow}t]y \mapsto_{\mathtt{m}} z[x{\leftarrow}y][z{\leftarrow}t]$.

The *exponential rule* $\mapsto_{\mathtt{e}_W}$ replaces a single variable occurrence, the one appearing in the context W. Example: $(xx)[x{\leftarrow}y[y{\leftarrow}t]] \mapsto_{\mathtt{e}_W} (x(y[y{\leftarrow}t]))[x{\leftarrow}y[y{\leftarrow}t]]$. The notation $\mapsto_{\mathtt{e}_W}$ stresses that the rule is parametric in a notion of context W, that specifies where the variable replacements are allowed, and which shall be exploited to define the CbS strategy in Sect. 8.

The *garbage collection (GC) rule by value* $\mapsto_{\mathtt{gcv}}$ eliminates a value v, keeping the list of substitutions S previously surrounding the value, which might contain terms that are not values, and so cannot be erased with v. This rule is the CbV erasing rule. Example: $(\lambda z.yy)[x{\leftarrow}\mathtt{I}[w{\leftarrow}t]] \mapsto_{\mathtt{gcv}} (\lambda z.yy)[w{\leftarrow}t]$, where $\mathtt{I}$ is the identity.

The three root rules $\mapsto_{\mathtt{m}}$, $\mapsto_{\mathtt{e}_W}$, and $\mapsto_{\mathtt{gcv}}$ are then closed by weak contexts. We shorten $W\langle t\rangle \to_{\mathtt{wm}} W\langle s\rangle$ if $t \mapsto_{\mathtt{m}} s$ with $\to_{\mathtt{wm}} := W\langle \mapsto_{\mathtt{m}}\rangle$, and similarly for the other rules. The reduction $\to_{\mathtt{w}}$ encompasses all possible reductions in the weak SSC.

Silliness Check and the Silly Extra Copy. Let us evaluate with the SSC the examples used in the introduction to explain silly and wise behaviour. About silly erasures, $(\lambda x.\mathtt{I})\Omega$ diverges, as in CbV. One can reduce it to $\mathtt{I}[x{\leftarrow}\Omega]$, but then there is no way of erasing Ω, which is not a value and does not reduce to a value. About silly duplications, $(\lambda x.xx)(\mathtt{II})$ can reduce both to $\mathtt{II}(\mathtt{II})$ and to $(\lambda x.xx)\mathtt{I}$, as in CbN. The silly aspect is the fact that one *can* reduce to $\mathtt{II}(\mathtt{II})$, since in CbV this is *not* possible. The CbS strategy of Sect. 8 shall always select the silly option.

Note that $(\lambda x.xx)(\mathtt{II})$ reduces to $(xx)[x{\leftarrow}\mathtt{II}]$ and then *three* copies of $\mathtt{II}$ can be evaluated, as one can reduce to $\mathtt{II}(\mathtt{II})[x{\leftarrow}\mathtt{II}]$ and the copy in the ES has to be evaluated before being erased, because it is not a value. Such a further copy would not be evaluated in CbN, as terms are erased without being evaluated. We refer to this aspect as to the *silly extra copy*.

3 Basic Rewriting Notions

Given a rewriting relation $\to_{\mathsf{r}}$, we write $d\colon t \to_{\mathsf{r}}^* s$ for a $\to_{\mathsf{r}}$-reduction sequence from t to s, the length of which is noted $|d|$. Moreover, we use $|d|_a$ for the number of a-*steps* in d, for a sub-relation $\to_a$ of $\to_{\mathsf{r}}$.

A term t is *weakly* r-*normalizing*, noted $t \in \mathrm{WN}_{\mathsf{r}}$, if $d\colon t \to_{\mathsf{r}}^* s$ with s r-normal; and t is *strongly* r-*normalizing*, noted $t \in \mathrm{SN}_{\mathsf{r}}$, if there are no diverging r-sequences from t, or, equivalently, if all its reducts are in SN_{r}.

According to Dal Lago and Martini [29], a relation $\to_{\mathsf{r}}$ is *diamond* if $s_1 \,{}_{\mathsf{r}}{\leftarrow} t \to_{\mathsf{r}} s_2$ and $s_1 \neq s_2$ imply $s_1 \to_{\mathsf{r}} u \,{}_{\mathsf{r}}{\leftarrow} s_2$ for some u. If $\to_{\mathsf{r}}$ is diamond then:

1. *Confluence*: $\to_{\mathsf{r}}$ is confluent, that is, $s_1 \,{}_{\mathsf{r}}^*{\leftarrow} t \to_{\mathsf{r}}^* s_2$ implies $s_1 \to_{\mathsf{r}}^* u \,{}_{\mathsf{r}}^*{\leftarrow} s_2$ for some u;
2. *Length invariance*: all r-evaluations to normal form with the same start term have the same length (i.e. if $d\colon t \to_{\mathsf{r}}^* s$ and $d'\colon t \to_{\mathsf{r}}^* s$ with s $\to_{\mathsf{r}}$-normal then $|d| = |d'|$);
3. *Uniform normalization*: t is weakly r-normalizing if and only if it is strongly r-normalizing.

Basically, the diamond captures a more liberal form of determinism.

4 Rewriting Properties

In this section, we study some rewriting properties of the weak SSC. We give a characterization of weak normal forms and prove postponement of garbage collection by value and confluence.

Characterization of Normal Forms. The characterization of (possibly open) weak normal forms requires the following concept.

▶ **Definition 1** (Shallow free variables). *The set* $\mathtt{shfv}(t)$ *of* shallow free variables *of* t *is the set of variables with occurrences out of abstractions in* t*:*

$$\begin{array}{lcllcl} \mathtt{shfv}(x) & := & \{x\} & \mathtt{shfv}(\lambda x.t) & := & \emptyset \\ \mathtt{shfv}(ts) & := & \mathtt{shfv}(t) \cup \mathtt{shfv}(s) & \mathtt{shfv}(t[x{\leftarrow}s]) & := & (\mathtt{shfv}(t) \setminus \{x\}) \cup \mathtt{shfv}(s) \end{array}$$

▶ **Proposition 2.** *t is $\to_{\mathtt{w}}$-normal if and only if t is a weak normal term according to the following grammar:*

$$\begin{array}{rrcll} \textsc{Weak answers} & a, a & ::= & v & \mid a[x{\leftarrow}i] \text{ with } x \notin \mathtt{shfv}(a) \\ & & & & \mid a[x{\leftarrow}a'] \text{ with } x \in \mathtt{fv}(a) \setminus \mathtt{shfv}(a) \\ \textsc{Inert terms} & i, i' & ::= & x \mid in & \mid i[x{\leftarrow}i'] \text{ with } x \notin \mathtt{shfv}(i) \\ & & & & \mid i[x{\leftarrow}a] \text{ with } x \in \mathtt{fv}(i) \setminus \mathtt{shfv}(i) \\ \textsc{Weak normal terms} & n, n' & ::= & a \mid i & \end{array}$$

Postponement of GC by Value. As it is usually the case in all the dialects of the LSC, the erasing rule of the weak SSC, that is $\to_{\mathtt{wgcv}}$, can be postponed. Let $\to_{\mathtt{w}\neg\mathtt{gcv}} := \to_{\mathtt{wm}} \cup \to_{\mathtt{we}}$.

▶ **Proposition 3** (Postponement of garbage collection by value). *If $d : t \to_{\mathtt{w}}^{*} s$ then $t \to_{\mathtt{w}\neg\mathtt{gcv}}^{k} \to_{\mathtt{wgcv}}^{h} s$ with $k = |d|_{\mathtt{w}\neg\mathtt{gcv}}$ and $h \geq |d|_{\mathtt{wgcv}}$.*

Local Termination. To prepare for confluence, we recall the following crucial property, which is essentially inherited from the LSC, given that termination for $\to_{\mathtt{wgcv}}$ is trivial.

▶ **Proposition 4** (Local termination). *Reductions $\to_{\mathtt{wm}}$, $\to_{\mathtt{we}}$, and $\to_{\mathtt{wgcv}}$ are strongly normalizing separately.*

Confluence. The proof of confluence for the weak SSC given here is a minor variant of what would be done for the LSC, except that there is no direct proof of confluence in the literature for the LSC[1]. Overviewing the proof allows us to explain a design choice of the SSC.

The proof is based on an elegant technique resting on local diagrams and local termination, namely the Hindley-Rosen method. In our case, it amounts to prove that the three rules $\to_{\mathtt{wm}}$, $\to_{\mathtt{we}}$, and $\to_{\mathtt{wgcv}}$ are confluent separately, proved by local termination and Newman's lemma, and commute, proved by local termination and Hindley's strong commutation. Confluence then follows by Hindley-Rosen lemma, for which the union of confluent and commuting reductions is confluent. The Hindley-Rosen method is a modular technique often used for confluence of extensions of the λ-calculus, for instance in [20, 34, 51, 27, 50, 26, 18, 16, 4].

▶ **Lemma 5** (Local confluence). *Reductions $\to_{\mathtt{wm}}$ and $\to_{\mathtt{wgcv}}$ are diamond, $\to_{\mathtt{we}}$ is locally confluent.*

[1] Confluence of the LSC holds, as it follows from stronger results about residuals in Accattoli et al. [8] but here we want to avoid the heaviness of residuals.

In contrast to confluence, commutation of two reductions $\to_1$ and $\to_2$ does *not* follow from their *local* commutation and strong normalization. In our case, however, the rules verify a non-erasing form of Hindley's strong (local) commutation [36] of $\to_1$ over $\to_2$, here dubbed *strict strong commutation*: if $s_1\ {}_1\!\leftarrow t \to_2 s_2$ then $\exists u$ such that $s_1 \to_2^+ u\ {}_1\!\leftarrow s_2$, that is, on one side of the commutation there are no duplications of steps, and on both sides there are no erasures, because $\to_{\mathtt{wgcv}}$ can only erase values, which do not contain redexes since evaluation is weak. Strong commutation and strong normalization do imply commutation.

▶ **Lemma 6** (Strict strong local commutations). *Reduction $\to_{\mathtt{we}}$ (resp. $\to_{\mathtt{we}}$; resp. $\to_{\mathtt{wgcv}}$) strictly strongly commutes over $\to_{\mathtt{wm}}$ (resp. $\to_{\mathtt{wgcv}}$; resp. $\to_{\mathtt{wm}}$).*

▶ **Theorem 7** (Confluence). *Reductions $\to_{\mathtt{w}}$ and $\to_{\mathtt{w}\neg\mathtt{gcv}}$ are confluent.*

Proof. By Newman lemma, local confluence (Lemma 5) and local termination (Prop. 4) imply that $\to_{\mathtt{wm}}$, $\to_{\mathtt{we}}$, and $\to_{\mathtt{wgcv}}$ are confluent separately. By a result of Hindley [36], strict strong local commutation (Lemma 6) and local termination imply that $\to_{\mathtt{wm}}$, $\to_{\mathtt{we}}$, and $\to_{\mathtt{wgcv}}$ are pairwise commuting. By Hindley-Rosen lemma, $\to_{\mathtt{w}} = \to_{\mathtt{wm}} \cup \to_{\mathtt{we}} \cup \to_{\mathtt{wgcv}}$ and $\to_{\mathtt{w}\neg\mathtt{gcv}} = \to_{\mathtt{wm}} \cup \to_{\mathtt{we}}$ are confluent. ◀

▶ Remark 8. The design choice that values are only abstraction is motivated by the commutation of $\to_{\mathtt{we}}$ over $\to_{\mathtt{wgcv}}$. Indeed, if one considers variables as values (thus allowing the erasure of variables by $\to_{\mathtt{gcv}}$), such a commutation fails, and confluence does not hold, as the following non-commuting (and non-confluent) span shows: $x[z\leftarrow ww]\ {}_{\mathtt{wgcv}}\!\leftarrow x[y\leftarrow z][z\leftarrow ww] \to_{\mathtt{e}_W} x[y\leftarrow ww][z\leftarrow ww]$.

Interestingly, something similar happens in CbNeed, where values are only abstractions too. Indeed, if one considers variables as values (thus allowing the duplication of variables in CbNeed), then one has the following non-closable critical pair for the CbNeed strategy, non-closable because y is not needed in $\mathtt{I}(\lambda z.wx)[x\leftarrow y][y\leftarrow \mathtt{I}]$ (the diagram closes in the CbNeed *calculus* but the CbNeed *strategy* is not confluent):

$$\begin{array}{ccc} x(\lambda z.wx)[x\leftarrow y][y\leftarrow \mathtt{I}] & \longrightarrow & x(\lambda z.wx)[x\leftarrow \mathtt{I}][y\leftarrow \mathtt{I}] \\ \downarrow & & \vdots \\ y(\lambda z.wx)[x\leftarrow y][y\leftarrow \mathtt{I}] & \dashrightarrow & \mathtt{I}(\lambda z.wx)[x\leftarrow \mathtt{I}][y\leftarrow \mathtt{I}] \end{array}$$

5 Silly Multi Types

In this section, we introduce multi types and the silly multi type system.

Inception. The design of the silly type system is specular to the one for CbNeed by Accattoli et al. [12]. The CbNeed one tweaks the CbV system in the literature (due to Ehrhard [32]) by changing its rules for applications and ESs, which are the rules using multi-sets, as to accommodate CbN erasures. The silly one given here tweaks the CbN system in the literature (due to de Carvalho [31]) by changing the same rules to accommodate CbV erasures. In both cases, the underlying system is responsible for the duplication behavior. The desired erasure behavior is then enforced by changing the rules using multi-sets.

The silly system is in Fig. 2. It is the variant of the system for CbN in [12] (itself a reformulation of [31] tuned for weak evaluation) obtained by adding "$\uplus$[norm]" in the right premise of both rules @ and ES. Such an extra typing for these rules captures the *silly extra copy* mentioned at the end of Sect. 2. More details are given at the end of this section.

$$\begin{array}{rrcl} \text{Linear types} & L, L' & ::= & \mathtt{norm} \mid M \to L \\ \text{Multi types} & M, N & ::= & [L_i]_{i\in I} \text{ where } I \text{ is a finite set} \\ \text{Generic types} & T, T' & ::= & L \mid M \end{array}$$

$$\frac{}{x : [L] \vdash^{(0,1)} x : L}\ \mathsf{ax} \qquad \frac{(\Gamma_i \vdash^{(m_i,e_i)} t : L_i)_{i\in I}}{\uplus_{i\in I}\Gamma_i \vdash^{(\sum_{i\in I} m_i, \sum_{i\in I} e_i)} t : [L_i]_{i\in I}}\ \mathsf{many}$$

$$\frac{}{\vdash^{(0,0)} \lambda x.t : \mathtt{norm}}\ \mathsf{ax}_\lambda \qquad \frac{\Gamma \vdash^{(m,e)} t : M \to L \qquad \Delta \vdash^{(m',e')} s : M \uplus [\mathtt{norm}]}{\Gamma \uplus \Delta \vdash^{(m+m'+1, e+e')} ts : L}\ @$$

$$\frac{\Gamma \vdash^{(m,e)} t : L}{\Gamma \setminus\!\setminus x \vdash^{(m,e)} \lambda x.t : \Gamma(x) \to L}\ \lambda \qquad \frac{\Gamma \vdash^{(m,e)} t : L \qquad \Delta \vdash^{(m',e')} s : \Gamma(x) \uplus [\mathtt{norm}]}{(\Gamma \setminus\!\setminus x) \uplus \Delta \vdash^{(m+m', e+e')} t[x \leftarrow s] : L}\ \mathsf{ES}$$

Figure 2 The silly multi type system.

CbS Types and Judgements. *Linear types* and *multi(-sets) types* are defined by mutual induction in Fig. 2. Note the linear constant $\mathtt{norm}$ used to type abstractions, which are normal terms. For conciseness, sometimes we shorten it to $\mathtt{n}$. We shall show that every normalizing term is typable with $\mathtt{norm}$, hence its name. The constant $\mathtt{norm}$ shall also play a role in our quantitative study in Sect. 9. The empty multi set $[\,]$ is also noted $\mathbf{0}$.

A multi type $[L_1, \dots, L_n]$ has to be intended as a conjunction $L_1 \wedge \cdots \wedge L_n$ of linear types $L_1, \dots, L_n$, for a commutative, associative, non-idempotent conjunction $\wedge$ (morally a tensor $\otimes$), of neutral element $\mathbf{0}$. The intuition is that a linear type corresponds to a single use of a term t, and that t is typed with a multiset M of n linear types if it is going to be used (at most) n times, that is, if t is part of a larger term s, then a copy of t shall end up in evaluation position during the evaluation of s.

Judgments have shape $\Gamma \vdash^{(m,e)} t : T$ where t is a term, m and e are two natural numbers, T is either a multi type or a linear type, and Γ is a *type context*, i.e., a total function from variables to multi types such that $\mathtt{dom}(\Gamma) := \{x \mid \Gamma(x) \neq \mathbf{0}\}$ is finite, usually written as $x_1 : M_1, \dots, x_n : M_n$ (with $n \in \mathbb{N}$) if $\mathtt{dom}(\Gamma) \subseteq \{x_1, \dots, x_n\}$ and $\Gamma(x_i) = M_i$ for $1 \leq i \leq n$.

The indices m and e shall be used for measuring the length of evaluation sequences via type derivations, namely to measure the number of $\to_{\mathtt{wm}}$ and $\to_{\mathtt{we}}$ steps. There is no index for $\to_{\mathtt{wgcv}}$ steps in order to stay close to the type systems in Accattoli et al. [12] for CbN/CbV/CbNeed, that do not have an index for GC either; the reason being that GC (by value) can be postponed. A quick look to the typing rules shows that m and e are not really needed, as m can be recovered as the number of $\mathtt{app}$ rules, and e as the number of $\mathtt{ax}$ rules. It is however handy to note them explicitly.

We write $\Gamma \vdash t : T$ when the information given by m and e is not relevant.

Typing Rules. The abstraction rule λ uses the notation $\Gamma \setminus\!\setminus x$ for the type context defined as Γ on every variable but possibly x, for which $(\Gamma \setminus\!\setminus x)(x) = \mathbf{0}$. It is a compact way to express the rule in both the cases $x \in \mathtt{dom}(\Gamma)$ and $x \notin \mathtt{dom}(\Gamma)$.

Rules @ and ES require the argument to be typed with $M \uplus [\mathtt{norm}]$, which is necessarily introduced by rule many, the hypotheses of which are a multi set of derivations, indexed by a possibly empty set I. When I is empty, the rule has one premises, the one for $\mathtt{norm}$.

Type Derivations. We write $\pi \triangleright \Gamma \vdash t : L$ if π is a (*type*) *derivation* (i.e. a tree constructed using the rules in Fig. 2) of final judgment $\Gamma \vdash t : L$.

The *size* $|\pi|$ of a derivation $\pi \triangleright \Gamma \vdash^{(m,e)} t : A$ is the number of non-many rules, which is always greater or equal to the sum of the indices, that is, $|\pi| \geq m + e$.

Further Technicalities about Types. The type context Γ is *empty* if $\mathtt{dom}(\Gamma) = \emptyset$, and we write $\vdash t : L$ when Γ is empty. *Multi-set sum* $\uplus$ is extended to type contexts point-wise, i.e. $(\Gamma \uplus \Delta)(x) := \Gamma(x) \uplus \Delta(x)$ for each variable x. This notion is extended to a finite family of type contexts as expected, in particular $\uplus_{i\in J}\Gamma_i$ is the empty context when $J = \emptyset$. Given two type contexts Γ and Δ such that $\mathtt{dom}(\Gamma) \cap \mathtt{dom}(\Delta) = \emptyset$, the type context Γ, Δ is defined by $(\Gamma, \Delta)(x) := \Gamma(x)$ if $x \in \mathtt{dom}(\Gamma)$, $(\Gamma, \Delta)(x) := \Delta(x)$ if $x \in \mathtt{dom}(\Delta)$, and $(\Gamma, \Delta)(x) := \mathbf{0}$ otherwise. Note that $\Gamma, x : \mathbf{0} = \Gamma$, where we implicitly assume $x \notin \mathtt{dom}(\Gamma)$.

Relevance. Note that no weakening is allowed in axioms. An easy induction then shows:

▶ **Lemma 9** (Type contexts and variable occurrences). *Let* $\pi \triangleright \Gamma \vdash^{(m,e)} t : L$ *be a derivation. Then* $\mathtt{shfv}(t) \subseteq \mathtt{dom}(\Gamma) \subseteq \mathtt{fv}(t)$.

Lemma 9 implies that derivations of closed terms have empty type context. Note that free variables of t might not be in $\mathtt{dom}(\Gamma)$, if they only occurr in abstractions typed with ax_λ.

Typing the Silly Extra Copy. The empty multi type $\mathbf{0}$ is the type for variables that do not occur or whose occurrences are unreachable by weak evaluation. A typical example is $\lambda x.y$, that can be typed only with arrow types of the form $\mathbf{0} \to L$ (plus of course with $\mathtt{norm}$), because of Lemma 9. Note that in the silly system every term – even diverging ones – can be typed with $\mathbf{0}$ by rule many (taking 0 premises). In CbN, an argument for $\lambda x.y$ would only need to be typed with $\mathbf{0}$, as typability with $\mathbf{0}$ means that the term shall be erased. In the silly system, instead, the application rule @ requires an argument of $\lambda x.y$ to additionally be typed with $[\mathtt{norm}]$, because of the "$\uplus[\mathtt{norm}]$" requirement for arguments (and ESs), forcing the argument to be $\to_{\mathtt{w}}$ normalizing and capturing silly erasures.

Now, note that the modification $\uplus[\mathtt{norm}]$ at work in the silly system concerns *all* arguments and ESs, not only those associated to $\mathbf{0}$. This is what correspond at the type level to the reduction of the *silly extra copy* of *every* argument/ES (out of abstractions).

Relationship with the Literature. Our system is essentially the one used by Kesner and Ventura to measure CbN strong normalization in the LSC with respect to strong evaluation, i.e. possibly under abstraction [42]. There are two differences. Firstly, they tweak the @ and ES CbN rules with $\uplus[L]$ rather than with $\uplus[\mathtt{norm}]$, that is, they allow an arbitrary linear type for the additional copy to be evaluated. Secondly, they do not have rule ax_λ, because their evaluation is strong. Thus, $\mathtt{I}[x \leftarrow \lambda y.\Omega]$ is not typable in [42], while here it is.

The CbV system by Manzonetto et al. [46] is similar to ours in that it tweaks the CbN system and requires multi-sets to be non-empty.

6 The Weak Calculus, Types, and Strong Normalization

Here, we show that silly multi types characterize strong normalization in the weak SSC. The proof technique is standard: we prove correctness – i.e. t typable implies $t \in \mathrm{SN}_{\mathtt{w}}$ – via subject reduction, and completeness – i.e. $t \in \mathrm{SN}_{\mathtt{w}}$ implies t typable – via subject expansion and typability of normal forms. Note that SN in our weak case is simpler than SN in strong calculi, since here erasing steps cannot erase divergence. Actually, they cannot erase *any* step, as guaranteed by strict commutation in Sect. 4. At the end of the section, we shall indeed obtain *uniform normalization*, i.e. that weak and strong normalization for $\to_{\mathtt{w}}$ coincide.

Correctness. As it is standard, subject reduction for $\to_{\mathtt{we}}$ is based on a linear substitution lemma, in the technical report [14].

▶ **Proposition 10** (Quantitative subject reduction for Weak SSC). *Let $\pi \triangleright \Gamma \vdash^{(m,e)} t : L$ be a derivation.*

1. Multiplicative*: if $t \to_{\mathtt{wm}} s$ then $m \geq 1$ and there exists $\rho \triangleright \Gamma \vdash^{(m',e)} s : L$ with $m > m'$.*
2. Exponential*: if $t \to_{\mathtt{we}} s$ then $e \geq 1$ and there exists $\rho \triangleright \Gamma \vdash^{(m,e')} s : L$ with $e > e'$.*
3. GC by value*: if $t \to_{\mathtt{wgcv}} s$ then there exists $\rho \triangleright \Gamma \vdash^{(m,e)} s : L$ with $|\pi| > |\rho|$.*

Note that $\to_{\mathtt{wgcv}}$ steps do not change the m and e indices. This is a consequence of rules @ and ES having been modified for CbS with $\uplus[\mathtt{norm}]$ (and not with $\uplus[L]$ as in [42]).

▶ **Theorem 11** (Weak SSC correctness). *Let t be a term. If $\pi \triangleright \Gamma \vdash^{(m,e)} t : L$ then $t \in SN_{\mathtt{w}}$. Moreover, if $d : t \to_{\mathtt{w}}^{*} n$ is a normalizing sequence then $|d|_{\mathtt{wm}} \leq m$ and $|d|_{\mathtt{we}} \leq e$.*

Proof. By lexicographic induction on $(m + e, |\pi|)$ and case analysis on whether t reduces or not. If t is $\to_{\mathtt{w}}$-normal then the statement trivially holds. If t is not $\to_{\mathtt{w}}$-normal we show that all its reducts are SN for $\to_{\mathtt{w}}$, that is, t is SN. If $t \to_{\mathtt{wm}} u$ then by quantitative subject reduction (Prop. 28) there is a derivation $\rho \triangleright \Gamma \vdash^{(m',e)} u : L$ with $m' < m$. By *i.h.*, u is SN. If $t \to_{\mathtt{we}} u$ or $t \to_{\mathtt{wgcv}} u$ we reason similarly, looking at the e index for $\to_{\mathtt{we}}$ and to the size $|\rho|$ of the derivation for $\to_{\mathtt{wgcv}}$. The *moreover* part follows from the fact that $\to_{\mathtt{wm}}$ (resp. $\to_{\mathtt{we}}$) steps strictly decrease m (resp. e). ◀

Completeness. For completeness, we first need typability of weak normal forms. The two points of the next proposition are proved by mutual induction. The second point is stronger, as it has a universal quantification about linear types, crucial for the induction to go through.

▶ **Proposition 12** (Weak normal forms are typable).

1. *Let a be a weak answer. Then there exists $\pi \triangleright \Gamma \vdash^{(0,0)} a : \mathtt{norm}$ with $\mathtt{dom}(\Gamma) = \mathtt{shfv}(a)$.*
2. *Let i be an inert term. Then for any linear type L there exist a type context Γ such that $\mathtt{dom}(\Gamma) = \mathtt{shfv}(i)$ and a derivation $\pi \triangleright \Gamma \vdash^{(0,0)} i : L$.*

The rest of the proof of completeness is dual to the one for correctness, with an anti-substitution lemma (in the technical report [14]) needed for subject expansion for $\to_{\mathtt{we}}$. We omit the indices because for completeness they are irrelevant.

▶ **Proposition 13** (Subject expansion for Weak SSC). *Let $\pi \triangleright \Gamma \vdash s : L$ be a derivation. If $t \to_{\mathtt{w}} s$ then there exists a derivation $\rho \triangleright \Gamma \vdash t : L$.*

▶ **Theorem 14** (Weak SSC completeness). *Let t be a term. If $t \to_{\mathtt{w}}^{*} n$ and n is a weak normal form then there exists $\pi \triangleright \Gamma \vdash t : \mathtt{norm}$.*

Proof. By induction on $k = |d|$. If $k = 0$: then $t = n$ and is typable with $\mathtt{norm}$ by Prop. 12. If $k > 0$ then $t \to_{\mathtt{w}} u \to_{\mathtt{w}}^{k-1} n$ for some u and by *i.h.* there is a derivation $\pi' \triangleright \Gamma \vdash u : L$. By subject expansion (Prop. 13), $\pi \triangleright \Gamma \vdash t : L$. ◀

▶ **Corollary 15** (Weak SSC Uniform normalization). *t is weakly $\to_{\mathtt{w}}$-normalizing if and only if t is strongly $\to_{\mathtt{w}}$-normalizing.*

Proof. The non-obvious direction is $\Rightarrow$. By completeness (Th. 14), $t \in \mathrm{WN}_{\mathtt{w}}$ implies typability, which in turn, by correctness (Th. 11), implies $t \in \mathrm{SN}_{\mathtt{w}}$. ◀

Terms	t, s, u	$::=$	$x \mid \lambda x.t \mid ts$	Rewriting rule	
Values	v, v'	$::=$	$\lambda x.t$	$(\lambda x.t)v$	$\mapsto_{\beta_v} \quad t\{x \leftarrow v\}$
CbV contexts	V	$::=$	$\langle\cdot\rangle \mid tV \mid Vt$	$\rightarrow_{\beta_v}$	$:= \quad V\langle\mapsto_{\beta_v}\rangle$

Figure 3 The call-by-value λ-calculus.

7 Call-by-Value and Operational Equivalence

Here, we show that the silly multi types characterize CbV termination as well, and infer the operational equivalence of the SSC and CbV.

Closed CbV without ESs. We define the CbV λ-calculus in Fig. 3, mostly following the presentation of Dal Lago and Martini [29], for which the β_v-rule is non-deterministic but diamond (thus trivially uniformly normalizing, see the rewriting preliminaries). The only change with respect to [29] is that here values are only abstractions, for uniformity with the Weak SSC. We shall consider the weak evaluation of closed terms only, for which there is no difference whether variables are values or not, since free variables cannot be arguments (out of abstractions) anyway. Closed normal forms are exactly the abstractions.

We discuss only the closed case for CbV because the silly type system is *not* correct for CbV with open terms. This point is properly explained after the operational equivalence theorem. It is also the reason why we do not present CbV via the LSC, as also explained after the theorem. The problem with open terms does not hinder the operational equivalence of the SSC and CbV, because contextual equivalence is based on closed terms only.

Judgements for CbV. In this section, the index e of the silly type system does not play any role, because $\rightarrow_{\beta_v}$ steps are bound only by the m index. Therefore, we omit e and write $\pi \triangleright \Gamma \vdash^m t : L$ instead of $\pi \triangleright \Gamma \vdash^{(m,e)} t : L$.

Correctness. The proof technique is the standard one. As usual, subject reduction is proved via a substitution lemma specified in the technical report [14]. The index m is used as decreasing measure to prove correctness.

▶ **Proposition 16** (Quantitative subject reduction for Closed CbV). *Let t be a closed term. If $\pi \triangleright \vdash^m t : L$ and $t \rightarrow_{\beta_v} s$ then $m \geq 1$ and there exists $\rho \triangleright \vdash^{m'} s : L$ with $m > m'$.*

▶ **Theorem 17** (Closed CbV correctness). *Let t be a closed term. If $\pi \triangleright \vdash^m t : L$ then there are an abstraction v and a reduction sequence $d\colon t \rightarrow^*_{\beta_v} v$ with $|d| \leq m$.*

Completeness. Completeness is also proved in a standard way, omitting the index m because it is irrelevant.

▶ **Proposition 18** (Subject expansion for Closed CbV). *If $\pi \triangleright \vdash s : L$ and $t \rightarrow_{\beta_v} s$ then and there exists a typing ρ such that $\rho \triangleright \vdash t : L$.*

▶ **Theorem 19** (Closed CbV completeness). *Let t be a closed λ-term. If there exists a value v and a reduction sequence $d\colon t \rightarrow^*_{\beta_v} v$ then $\pi \triangleright \vdash t : \mathtt{norm}$.*

Operational Equivalence. We define abstractly contextual equivalence for a language of terms and arbitrary contexts which are terms with an additional hole construct.

▶ **Definition 20** (Contextual Equivalence). *Given a rewriting relation $\to$, we define the associated* contextual equivalence $\simeq_{\mathtt{c}}$ *as follows: $t \simeq_{\mathtt{c}} t'$ if, for all contexts C such that $C\langle t\rangle$ and $C\langle t'\rangle$ are closed terms, $C\langle t\rangle$ is weakly $\to$-normalizing iff $C\langle t'\rangle$ is weakly $\to$-normalizing.*

Let $\simeq_C^{\mathtt{silly}}$ and $\simeq_C^{\mathtt{value}}$ be the contextual equivalences for $\to_{\mathtt{w}}$ and $\to_{\beta_v}$. The next section shall show that $\simeq_C^{\mathtt{silly}}$ can equivalently be defined using the CbS strategy.

▶ **Theorem 21** (Operational equivalence of CbS and CbV). *On λ-terms, $t \simeq_C^{\mathtt{silly}} s$ iff $t \simeq_C^{\mathtt{value}} s$.*

Proof. On closed λ-terms, both $\to_{\mathtt{w}}$-termination and $\to_{\beta_v}$-termination are equivalent to typability in the silly system (Theorems 11 and 14 for $\to_{\mathtt{y}}$ and Theorems 17 and 19 for $\to_{\beta_v}$). Thus the contextual equivalences coincide. ◀

Call-by-Silly Helps to Prove Contextual Equivalence. As the last theorem says, contextual equivalences induced by CbV and CbS coincide. Even though they equate the same terms, CbS reduction sometimes provides a way to prove CbV contextual equivalence in cases where CbV does not. Consider the following four different terms, where i could be any normal form that is not of the shape $S\langle v\rangle$, for example $i = y\mathtt{I}$:

$$(\lambda x.xx)\, i \qquad (\lambda x.xi)\, i \qquad (\lambda x.ii)\, i \qquad ii$$

These four terms can intuitively be seen as CbV contextually equivalent, as we now outline. When one of these terms is plugged in a closing context C, reduction shall provide substitutions on i making it either converge to a value v or diverge. if it diverges, so will the four terms. If it converges to v, then all four terms will reduce to vv. This reasoning however cannot easily be made formal.

The easiest way to prove that two terms are contextually equivalent for a reduction $\to_r$ is to prove that they are related by $=_r$, the smallest equivalence relation including $\to_r$. We shall now see how the silly calculus helps in equating more terms (than CbV) with its reduction.

The four terms above are not $=_{\beta_v}$-related. First, note that the four terms are all $\to_{\beta_v}$-normal (assuming that i is $\to_{\beta_v}$-normal). If they were $=_{\beta_v}$-related, then by the Church-Rosser property they should have a common reduct. As the four terms are syntactically different normal forms, they cannot be equated by $=_{\beta_v}$.

The first three terms are $=_{\mathtt{w}}$-related. The first three terms rewrite in the SSC to $ii[x{\leftarrow}i]$, which is why $(\lambda x.xx)\, i =_{\mathtt{w}} (\lambda x.xi)\, i =_{\mathtt{w}} (\lambda x.ii)\, i =_{\mathtt{w}} ii[x{\leftarrow}i]$.

Unfortunately, the fourth term ii is a $\to_{\mathtt{w}}$-normal form and hence $ii \neq_{\mathtt{w}} ii[x{\leftarrow}i]$. Thus, not all CbV contextually equivalent terms are equated by the silly calculus, not even when the difference amounts to the duplication of a non-value term. The issue has to do with the *silly extra copy* that cannot be erased easily. We believe that a small-step silly calculus could help equating more terms, but we have not managed to work out multi types for such a calculus.

We refer the reader to the technical report [14] for the proof that $=_{\mathtt{w}}$-related terms are contextually equivalent.

The Issue with Open CbV and the Silly Type System. There is an issue if one considers the silly type system relatively to CbV with open terms, namely subject reduction breaks. This point is delicate. In fact, there are no issues if one considers only Plotkin's β_v rule, except that Plotkin's rule is *not* an adequate operational semantics for CbV with open terms, as it is well-known and discussed at length by Accattoli and Guerrieri [10, 11]. Adequate

operational semantics for Open CbV do extend Plotkin's. One such semantics is Carraro and Guerrieri's *shuffling calculus* [27], that extends β_v by adding some σ-rules. We briefly discuss it here; the last paragraph of this section shall explain because we prefer it to the CbV LSC for the explanation of the issue with open terms.

It turns out that one of the σ rules breaks subject reduction for the silly type system (while there are no problems if one considers instead Ehrhard's CbV multi types [32]), as we now show. Rule $\mapsto_{\sigma_3}$ is defined as follows:

$$z((\lambda x.y)s) \quad \mapsto_{\sigma_3} \quad (\lambda x.zy)s$$

For $n \geq 1$, we have the following derivation for the source term $z((\lambda x.y)s)$ in the silly type system:

$$\dfrac{\pi_z \triangleright \dots \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{y:[A]\vdash y:A}\,\mathsf{ax}}{y:[A]\vdash \lambda x.y:\mathbf{0}\to A}\,\lambda \qquad \dfrac{\pi_s \triangleright \Gamma \vdash s:\mathtt{norm}}{\Gamma \vdash s:[\mathtt{norm}]}\,\mathsf{many}}{(y:[A],\Gamma\vdash(\lambda x.y)s:A)_{i=1,\dots,n}}\,@ \qquad \dfrac{\dfrac{\dfrac{}{y:[\mathtt{norm}]\vdash y:\mathtt{norm}}\,\mathsf{ax}}{y:[\mathtt{norm}]\vdash \lambda x.y:\mathbf{0}\to\mathtt{norm}}\,\lambda \qquad \dfrac{\pi_s \triangleright \Gamma \vdash s:\mathtt{norm}}{\Gamma \vdash s:[\mathtt{norm}]}\,\mathsf{many}}{y:[\mathtt{norm}],\Gamma\vdash(\lambda x.y)s:\mathtt{norm}}\,@}{y:[A^n,\mathtt{norm}],\Gamma^{n+1}\vdash(\lambda x.y)s:[A^n,\mathtt{norm}]}\,\mathsf{many}}{z:[[A^n,\mathtt{norm}]\to B],y:[A^n,\mathtt{norm}],\Gamma^{n+1}\vdash z((\lambda x.y)s):B}\,@$$

where $\pi_z \triangleright \dots$ stands for:

$$\dfrac{}{\pi_z \triangleright z:[[A^n]\to B]\vdash z:[A^n]\to B}\,\mathsf{ax}$$

The target term $(\lambda x.zy)s$ of rule $\mapsto_{\sigma_3}$, instead, can only be typed as follows, the key point being that Γ^{n+1} is replaced by Γ:

$$\dfrac{\dfrac{\dfrac{}{z:[[A^n]\to B]\vdash z:[A^n]\to B}\,\mathsf{ax} \qquad \dfrac{\dfrac{}{(y:[A]\vdash y:A)_{i=1,\dots,n}}\,\mathsf{ax} \qquad \dfrac{}{y:[\mathtt{norm}]\vdash y:\mathtt{norm}}\,\mathsf{ax}}{y:[A^n,\mathtt{norm}]\vdash y:[A^n,\mathtt{norm}]}\,\mathsf{many}}{z:[[A^n]\to B],y:[A^n,\mathtt{norm}]\vdash zy:B}\,@}{z:[[A^n]\to B],y:[A^n,\mathtt{norm}]\vdash \lambda x.zy:\mathbf{0}\to B}\,\lambda \qquad \dfrac{\pi_s \triangleright \Gamma \vdash s:\mathtt{norm}}{\Gamma\vdash s:[\mathtt{norm}]}\,\mathsf{many}}{z:[[A^n]\to B],y:[A^n,\mathtt{norm}],\Gamma\vdash(\lambda x.zy)s:B}\,@$$

This counter-example adapt the counter-example given by Delia Kesner to subject reduction for the multi type system by Manzonetto et al. [46, 37], as reported in the long version on Arxiv of [11], which appeared after the publication of [46, 37], where there is no mention of this issue. The work in the present paper can be actually seen as a clarification of the failure of subject reduction for the system in [46, 37]. Essentially, the system in [46, 37] is a system for CbS, not for CbV, but is therein used to study CbV strong evaluation with possibly open terms, unaware that the system models a different evaluation mechanism.

We conjecture that the silly type system is adequate for Open CbV (that is, a term is silly typable if and only if it is CbV terminating) even if it is not invariant for Open CbV (that is, subject reduction does not hold).

Naturality of the Issue. A first reaction to the shown issue is to suspect that something is wrong or ad-hoc in our approach, especially given that the operational equivalence of CbN and CbNeed does not suffer of this issue, that is, the CbN multi type system is invariant for CbNeed evaluation of open terms. At high-level, however, the issue is natural and to be expected, as we now explain. The two systems of each pair CbN/CbNeed and CbS/CbV have different duplicating policies and the same erasing policy. The pair CbN/CbNeed has no restrictions on erasure, thus open normal forms have no garbage. Therefore, the different ways in which they duplicate garbage are not observable. For the pair CbS/CbV, instead, erasure is restricted to *values*, with the consequence that garbage has to be evaluated before possibly being erased, and that with open terms some garbage might never be erased. Thus, the different duplicating policies of CbS/CbV leave different amounts of non-erasable garbage in open normal forms, which is observable and changes the denotational semantics.

The Open CbV LSC. Another adequate formalism for Open CbV is the VSC, or its micro-step variant, the CbV LSC. We now discuss the CbV LSC, but everything we say applies also to the VSC.

The difference between the shuffling calculus and the CbV LSC is that the latter uses ESs and modifies the rewriting rules at a distance. In particular, duplication is done by the following exponential rule:

$$W\langle x\rangle[x{\leftarrow}S\langle v\rangle] \quad \to_{\mathtt{ve}} \quad S\langle W\langle v\rangle[x{\leftarrow}v]\rangle$$

Subject reduction breaks also for the CbV LSC, as we can show by adapting Kesner's counter-example. Consider the step $(zy')[y'{\leftarrow}\mathtt{I}[y{\leftarrow}s]] \to_{\mathtt{ve}} (z\mathtt{I})[y'{\leftarrow}\mathtt{I}][y{\leftarrow}s]$:

$$\dfrac{\pi_1 \rhd z : [[A^n]{\to}B],\, y' : [A^n] \vdash zy' : B \qquad \dfrac{\dfrac{\dfrac{\pi_s \rhd \Gamma \vdash s : \mathtt{norm}}{\Gamma \vdash s : [\mathtt{norm}]}\,\mathsf{many} \qquad \pi_{\mathtt{I}} \vdash \mathtt{I} : A}{(\Gamma \vdash \mathtt{I}[y{\leftarrow}s] : A)_{i=1,\dots,n}}\,\mathsf{ES} \qquad \dfrac{\dfrac{}{\vdash \mathtt{I} : \mathtt{norm}}\,\mathsf{ax}_\lambda \qquad \dfrac{\pi_s \rhd \Gamma \vdash s : \mathtt{norm}}{\Gamma \vdash s : [\mathtt{norm}]}\,\mathsf{many}}{\Gamma \vdash \mathtt{I}[y{\leftarrow}s] : \mathtt{norm}}\,\mathsf{ES}}{\Gamma^{n+1} \vdash \mathtt{I}[y{\leftarrow}s] : [A^n, \mathtt{norm}]}\,\mathsf{many}}{z : [[A^n]{\to}B],\, \Gamma^{n+1} \vdash (zy')[y'{\leftarrow}\mathtt{I}[y{\leftarrow}s]] : B}\,\mathsf{ES}$$

As for $\mapsto_{\sigma_3}$, the key point is that that Γ^{n+1} gets replaced by Γ in the typing of the reduct:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{z : [[A^n]{\to}B] \vdash z : [A^n]{\to}B}\,\mathsf{ax} \qquad \dfrac{(\pi_{\mathtt{I}} \vdash \mathtt{I} : A)_{i=1,\dots,n} \qquad \dfrac{}{\vdash \mathtt{I} : \mathtt{norm}}\,\mathsf{ax}_\lambda}{\vdash \mathtt{I} : [A^n, \mathtt{norm}]}\,\mathsf{many}}{z : [[A^n]{\to}B] \vdash z\mathtt{I} : B}\,\mathsf{ES} \qquad \dfrac{\dfrac{}{\vdash \mathtt{I} : \mathtt{norm}}\,\mathsf{ax}_\lambda}{\vdash \mathtt{I} : [\mathtt{norm}]}\,\mathsf{many}}{z : [[A^n]{\to}B] \vdash (z\mathtt{I})[y'{\leftarrow}\mathtt{I}] : B}\,\mathsf{ES} \qquad \dfrac{\pi_s \rhd \Gamma \vdash s : \mathtt{norm}}{\Gamma \vdash s : [\mathtt{norm}]}\,\mathsf{many}}{z : [[A^n]{\to}B],\, \Gamma \vdash (z\mathtt{I})[y'{\leftarrow}\mathtt{I}][y{\leftarrow}s] : B}\,\mathsf{ES}$$

What breaks it is the use of the substitution context S in $\to_{\mathtt{ve}}$, which can be seen as the analogous of rule $\mapsto_{\sigma_3}$ of the shuffling calculus. The difference is that while $\mapsto_{\sigma_3}$ is needed only for open terms in the shuffling calculus, rule $\to_{\mathtt{ve}}$ is used also for the evaluation of closed terms in the CbV LSC. This is why we preferred to avoid using the CbV LSC to study Closed CbV. In fact, one can prove that in the closed case the modification of rule $\to_{\mathtt{ve}}$ without S is enough to reach normal forms. But this fact needs a technical study and a theorem, which we preferred to avoid.

8 The Call-by-Silly Strategy

In this section, we define the CbS evaluation strategy as a sort of extension of the CbN one. We then start by recalling the CbN erasing rule and the CbN strategy.

CbN Erasure. The *garbage collection (GC) rule* $\mapsto_{\mathtt{gc}}$ in Fig. 4 eliminates the ES $t[x{\leftarrow}s]$ when the bound variable x does not occur in t. Rule $\mapsto_{\mathtt{gc}}$ is the CbN form of erasure and it is not part of the rules of the SSC; it is given here only to be able to define the CbN strategy. Example: $(\lambda z.yy)[x{\leftarrow}\mathtt{I}[w{\leftarrow}t]] \mapsto_{\mathtt{gc}} \lambda z.yy$.

Call-by-Name. In Fig. 4, we define the CbN strategy $\to_{\mathtt{n}}$ of the weak LSC, first appeared in Accattoli et al. [5]. The CbN strategy uses the CbN GC rule $\mapsto_{\mathtt{gc}}$. Note that CbN evaluation contexts N never enter into arguments or ESs. The CbN strategy is almost deterministic: rules $\to_{\mathtt{nm}}$ and $\to_{\mathtt{ne}}$ are deterministic and its erasing rule $\to_{\mathtt{ngc}}$ is non-deterministic but diamond; for instance $\mathtt{I}[z{\leftarrow}\mathtt{II}]_{\mathtt{ngc}}{\leftarrow}\mathtt{I}[y{\leftarrow}\mathtt{I}][z{\leftarrow}\mathtt{II}] \to_{\mathtt{ngc}} \mathtt{I}[y{\leftarrow}\mathtt{I}]$, and then $\mathtt{I}[z{\leftarrow}\mathtt{II}] \to_{\mathtt{ngc}} \mathtt{I}_{\mathtt{ngc}}{\leftarrow}\mathtt{I}[y{\leftarrow}\mathtt{I}]$. Moreover, $\to_{\mathtt{gc}}$ is postponable, i.e. if $t \to_{\mathtt{n}}^* s$ then $t \to_{\mathtt{n}\neg\mathtt{gc}}^* \to_{\mathtt{gc}}^* s$ where $\to_{\mathtt{n}\neg\mathtt{gc}} := \to_{\mathtt{nm}} \cup \to_{\mathtt{ne}}$, which is why it is often omitted from the micro-step presentation of CbN.

CALL-BY-NAME STRATEGY $\to_{\mathtt{n}}$

NAME CTXS $N, N' ::= \langle\cdot\rangle \mid Nt \mid N[x{\leftarrow}t]$	$\to_{\mathtt{nm}} := N\langle \mapsto_{\mathtt{m}} \rangle$
ROOT GC $t[x{\leftarrow}s] \mapsto_{\mathtt{gc}} t \quad \text{if } x \notin \mathtt{fv}(t)$	$\to_{\mathtt{ne}} := N\langle \mapsto_{\mathtt{e}_N} \rangle$
	$\to_{\mathtt{ngc}} := N\langle \mapsto_{\mathtt{gc}} \rangle$
	$\to_{\mathtt{n}} := \to_{\mathtt{nm}} \cup \to_{\mathtt{ne}} \cup \to_{\mathtt{ngcv}}$

CALL-BY-SILLY STRATEGY $\to_{\mathtt{y}}$

ANSWERS $a, a' ::= v \mid a[x{\leftarrow}a']$	$\to_{\mathtt{ym}} := Y\langle \mapsto_{\mathtt{m}} \rangle$
AUX. CTXS $A, A' ::= \langle\cdot\rangle \mid a[x{\leftarrow}A] \mid A[x{\leftarrow}t]$	$\to_{\mathtt{ygcv}} := Y\langle \mapsto_{\mathtt{gcv}} \rangle$
SILLY CTXS $Y, Y' ::= A\langle N \rangle$	$\to_{\mathtt{ye_{AY}}} := A\langle \mapsto_{\mathtt{e}_Y} \rangle$
	$\to_{\mathtt{ye_{YN}}} := Y\langle \mapsto_{\mathtt{e}_N} \rangle$
	$\to_{\mathtt{y}} := \to_{\mathtt{ym}} \cup \to_{\mathtt{ye_{AY}}} \cup \to_{\mathtt{ye_{YN}}} \cup \to_{\mathtt{ygcv}}$

Figure 4 The call-by-name and call-by-silly strategies.

The Call-by-Silly Strategy. The CbS strategy $\to_{\mathtt{y}}$ is defined in Fig. 4 via silly evaluation contexts Y – explained next – (we use Y for *sillY* because S is already used for substitution contexts) and the CbV erasing rule $\mapsto_{\mathtt{gcv}}$ of the SSC. Letting GC aside, CbS is the extension of CbN that, once (non-erasing) CbN can no longer reduce, starts evaluating the ESs (out of abstractions) left hanging by the CbN strategy. Such an extension is specified via the *auxiliary contexts* A, whose key production is $t[x{\leftarrow}A]$ for evaluation contexts where t is a CbS normal form: evaluation enters an ESs only when it is sure that it is no longer needed. There are, however, a few subtleties.

Firstly, *no longer needed* does not necessarily mean *without free occurrences*, as occurrences might be blocked by abstractions, as for instance in $(\lambda y.x)[x{\leftarrow}s]$ where x occurs but only under abstraction, and the CbS has to evaluate inside s. Therefore, *no longer needed* rather means *without shallow free occurrences* (Def. 1, page 7).

Secondly, in order to be sure that no occurrences of x shall become shallow because of other steps, we shall ask that in $t[x{\leftarrow}Y]$ the sub-term t is a normal form, otherwise in a case such as $((\lambda y.x)\mathtt{I})[x{\leftarrow}s]$ the reduction of the multiplicative step to $x[y{\leftarrow}\mathtt{I}][x{\leftarrow}s]$ turns a blocked occurrence of x into a shallow occurrence. Because of rules at a distance, the term $(\lambda y.x)[x{\leftarrow}s]\mathtt{I}$ suffers of the same problem but the argument comes *on the right* of the ES. For this reason, we forbid the evaluation context to be applied once it enters an ES, that is, silly contexts Y are defined as $A\langle N\rangle$, that is, rigidly separating the construction that applies (at work in N) from the one that enters ES (at work in A).

Thirdly, such a rigid separation in the definition of silly contexts forces us to have *two* exponential rules. Exponential rules use contexts twice in their definition: to select the occurrence to replace in the root rule *and* to extend the applicability of the root rule. The subtlety is that if both these contexts are silly, then one can nest an A context (selecting an occurrence) under a N context (extending the rule), which, as explained, has to be avoided. Therefore there are two silly exponential rules $\to_{\mathtt{ye_{AY}}}$ and $\to_{\mathtt{ye_{YN}}}$ carefully designed as to avoid the dangerous combination (that would be given by $N\langle \mapsto_{\mathtt{e}_A} \rangle$).

Fourthly, in order to mimic the property of CbN that GC is postponable, we shall ask that in $t[x{\leftarrow}Y]$ the sub-term t is a normal form only for the *non-erasing* CbS, otherwise in a case such as $x[y{\leftarrow}\mathtt{I}][x{\leftarrow}s]$ the CbS strategy would be forced to erase $[y{\leftarrow}\mathtt{I}]$ before evaluating s. Normal forms for non-erasing CbS are characterized below (Prop. 23) exactly as the answers defined in Fig. 4. We shall also prove that $\to_{\mathtt{ygcv}}$ steps are postponable (Prop. 3).

The CbS strategy is supposed to be applied to closed terms only, while there is no closure hypothesis on the weak calculus.

▶ **Example 22.** A good example to observe the differences between CbN and CbS is given by the term $t := (\lambda y.\lambda x.x(\lambda w.x))\Omega(\mathtt{II})$ where $\mathtt{I} := \lambda z.z$, $\Omega := \delta\delta$, and $\delta := \lambda z.zz$. In CbN, it evaluates with 4 multiplicative steps and 3 exponential steps, as follows:

$$\begin{array}{lll} t & \to_{\mathtt{nm}} (\lambda x.x(\lambda w.x))[y{\leftarrow}\Omega](\mathtt{II}) & \to_{\mathtt{nm}} (x(\lambda w.x))[x{\leftarrow}\mathtt{II}][y{\leftarrow}\Omega] \\ & \to_{\mathtt{ne}} ((\mathtt{II})(\lambda w.x))[x{\leftarrow}\mathtt{II}][y{\leftarrow}\Omega] & \to_{\mathtt{nm}} (z[z{\leftarrow}\mathtt{I}](\lambda w.x))[x{\leftarrow}\mathtt{II}][y{\leftarrow}\Omega] \\ & \to_{\mathtt{ne}} (\mathtt{I}[z{\leftarrow}\mathtt{I}](\lambda w.x))[x{\leftarrow}\mathtt{II}][y{\leftarrow}\Omega] & \to_{\mathtt{nm}} z'[z'{\leftarrow}\lambda w.x][z{\leftarrow}\mathtt{I}][x{\leftarrow}\mathtt{II}][y{\leftarrow}\Omega] \\ & \to_{\mathtt{ne}} (\lambda w.x)[z'{\leftarrow}\lambda w.x][z{\leftarrow}\mathtt{I}][x{\leftarrow}\mathtt{II}][y{\leftarrow}\Omega] & =: s \end{array}$$

The obtained term s is normal for the multiplicative and exponential rules. Since in CbN one can erase every term, s then reduces (in CbN) as follows:

$$s \quad \to_{\mathtt{ngc}} \quad (\lambda w.x)[z{\leftarrow}\mathtt{I}][x{\leftarrow}\mathtt{II}][y{\leftarrow}\Omega] \quad \to_{\mathtt{ngc}} \quad (\lambda w.x)[x{\leftarrow}\mathtt{II}][y{\leftarrow}\Omega] \quad \to_{\mathtt{ngc}} \quad (\lambda w.x)[x{\leftarrow}\mathtt{II}]$$

The obtained term is now normal for the CbN strategy, since evaluation does not enter into ESs. Note that t diverges for CbV evaluation (defined in Sect. 7). In CbS, the first part of the evaluation of t, up to s, is the same as for CbN (where $\to_{\mathtt{ne}}$ steps become $\to_{\mathtt{ye_{YN}}}$ steps). Then, s evaluates differently than in CbN, diverging:

$$\begin{array}{llll} s & \to_{\mathtt{ygcv}} (\lambda w.x)[z{\leftarrow}\mathtt{I}][x{\leftarrow}\mathtt{II}][y{\leftarrow}\Omega] & \to_{\mathtt{ygcv}} (\lambda w.x)[x{\leftarrow}\mathtt{II}][y{\leftarrow}\Omega] & \\ & \to_{\mathtt{ym}} (\lambda w.x)[x{\leftarrow}z[z{\leftarrow}\mathtt{I}]][y{\leftarrow}\Omega] & \to_{\mathtt{ye_{AY}}} (\lambda w.x)[x{\leftarrow}\mathtt{I}[z{\leftarrow}\mathtt{I}]][y{\leftarrow}\Omega] & \\ & \to_{\mathtt{ygcv}} (\lambda w.x)[x{\leftarrow}\mathtt{I}][y{\leftarrow}\Omega] & \to_{\mathtt{ym}} (\lambda w.x)[x{\leftarrow}\mathtt{I}][y{\leftarrow}zz[z{\leftarrow}\delta]] & \ldots \end{array}$$

Rewriting Properties of CbS. We start by characterizing normal form for both the CbS strategy and its non-erasing variant $\to_{\mathtt{y}\neg\mathtt{gcv}} := \to_{\mathtt{ym}} \cup \to_{\mathtt{ye_{AY}}} \cup \to_{\mathtt{ye_{YN}}}$.

▶ **Proposition 23.** *Let t be closed.*

1. *t is $\to_{\mathtt{w}\neg\mathtt{gcv}}$-normal if and only if t is $\to_{\mathtt{y}\neg\mathtt{gcv}}$-normal if and only if t is an answer.*
2. *t is $\to_{\mathtt{w}}$-normal if and only if t is $\to_{\mathtt{y}}$-normal if and only if t is a strict answer:*

$$\textsc{Strict answers} \quad a_s \quad ::= \quad v \mid a_s[x{\leftarrow}a'_s] \quad \text{with } x \in \mathtt{fv}(a_s)$$

Next, we prove that the CbS strategy is almost deterministic. Its non-erasing rules are deterministic, while $\to_{\mathtt{ygcv}}$ is diamond. For instance:
$\mathtt{I}[z{\leftarrow}\delta]\,{}_{\mathtt{ygcv}}{\leftarrow}\mathtt{I}[y{\leftarrow}\mathtt{I}][z{\leftarrow}\delta] \to_{\mathtt{ygcv}} \mathtt{I}[y{\leftarrow}\mathtt{I}]$, and then $\mathtt{I}[z{\leftarrow}\delta] \to_{\mathtt{ygcv}} \mathtt{I}\,{}_{\mathtt{ygcv}}{\leftarrow}\mathtt{I}[y{\leftarrow}\mathtt{I}]$.

▶ **Proposition 24.** *Reductions $\to_{\mathtt{ym}}$, $\to_{\mathtt{ye_{AY}}}$, and $\to_{\mathtt{ye_{YN}}}$ are deterministic, and $\to_{\mathtt{ygcv}}$ is diamond.*

Lastly, we prove the *linear* postponement of $\to_{\mathtt{ygcv}}$, that is, the fact that it can be postponed while preserving the length of the evaluation. The length preservation shall be used to prove the maximality of $\to_{\mathtt{y}}$ in the next section.

▶ **Proposition 25** (Postponement of $\to_{\mathtt{ygcv}}$). *If $d : t \to_{\mathtt{y}}^* s$ then $t \to_{\mathtt{y}\neg\mathtt{gcv}}^k \to_{\mathtt{ygcv}}^h s$ with $k = |d|_{\mathtt{y}\neg\mathtt{gcv}}$ and $h = |d|_{\mathtt{ygcv}}$.*

9 Tight Derivations, Exact Lengths, and Maximality

Here, we focus on the CbS strategy on closed terms and isolate a class of *tight* type derivations whose indices measure exactly the number of non-erasing CbS steps, mimicking faithfully what was done in call-by-name/value/need by Accattoli et al. in [12]. An outcome shall be that the CbS strategy actually performs the *longest* weak evaluation on closed terms.

Tight Derivations. The basic tool for our quantitative analysis are tight type derivations, which shall be used to refine the correctness theorem of the weak case (Th. 11) in the case of closed terms. Tight derivations are defined via a predicate on their last judgement, as in [12].

▶ **Definition 26** (Tight types and derivations). *A type* T *is* tight *if* $T = \mathtt{norm}$ *or* $T = [\mathtt{norm}]$. *A derivation* $\pi \triangleright \Gamma \vdash^{(m,e)} t : T$ *is* tight *if* T *is tight.*

Tight Correctness. We first show that all tight derivations for answers have null indices.

▶ **Proposition 27** (Tight typing of normal forms for non-erasing CbS). *Let* a *be an answer and* $\pi \triangleright \Gamma \vdash^{(m,e)} a : \mathtt{norm}$ *be a derivation. Then* Γ *is empty and* $m = e = 0$.

Next, we refine subject reduction via tight derivations. The key point is that now the indices decrease of *exactly one* at each step. Tight correctness then follows.

▶ **Proposition 28** (Tight subject reduction for non-erasing CbS). *Let* $\pi \triangleright \Gamma \vdash^{(m,e)} t : \mathtt{norm}$ *be a tight derivation.*

1. Multiplicative: *if* $t \to_{\mathtt{ym}} s$ *then* $m \geq 1$ *and there is* $\rho \triangleright \Gamma \vdash^{(m-1,e)} s : \mathtt{norm}$.
2. Exponential: *if* $t \to_{\mathtt{ye_{AY}}} s$ *or* $t \to_{\mathtt{ye_{YN}}} s$ *then* $e \geq 1$ *and there is* $\rho \triangleright \Gamma \vdash^{(m,e-1)} s : \mathtt{norm}$.

▶ **Theorem 29** (Tight correctness for CbS). *Let* t *be a closed term and* $\pi \triangleright \vdash^{(m,e)} t : \mathtt{norm}$ *be a tight derivation. Then there is a weak normal form* n *such that* $d : t \to_{\mathtt{y}}^{*} n$ *with* $|d|_{\mathtt{ym}} = m$ *and* $|d|_{\mathtt{ye_{AY}},\mathtt{ye_{YN}}} = e$.

Proof. The proof is exactly as for weak correctness (Th. 11), except that if t is normal then the fact that $m = e = 0$ follows from Prop. 27 and that if t is not normal the equality on the number of steps is obtained by using tight subject reduction (Prop. 28) instead of the quantitative one. ◀

For tight completeness for closed terms, it is enough to observe that the statement of weak completeness (Th. 14) already gives a tight derivation $\Gamma \vdash t : \mathtt{norm}$, the type context Γ of which is empty because t is closed (Lemma 9).

▶ **Example 30.** We illustrate the tightness of multi types for CbS with an example. Consider the term $(\lambda y.yy)(\mathtt{II})$, the CbS evaluation of which is as follows (the first part coincides with the CbN evaluation):

CbN evaluation:

$(\lambda y.yy)(\mathtt{II})$

$\to_{\mathtt{ym}}$	$yy[y{\leftarrow}\mathtt{II}]$	$\to_{\mathtt{ye_{YN}}}$	$(\mathtt{II})y[y{\leftarrow}\mathtt{II}]$
$\to_{\mathtt{ym}}$	$(x[x{\leftarrow}\mathtt{I}])y[y{\leftarrow}\mathtt{II}]$	$\to_{\mathtt{ye_{YN}}}$	$(\mathtt{I}[x{\leftarrow}\mathtt{I}])y[y{\leftarrow}\mathtt{II}]$
$\to_{\mathtt{ym}}$	$z[z{\leftarrow}y][x{\leftarrow}\mathtt{I}][y{\leftarrow}\mathtt{II}]$	$\to_{\mathtt{ye_{YN}}}$	$y[z{\leftarrow}y][x{\leftarrow}\mathtt{I}][y{\leftarrow}\mathtt{II}]$
		$\to_{\mathtt{ye_{YN}}}$	$\mathtt{II}[z{\leftarrow}y][x{\leftarrow}\mathtt{I}][y{\leftarrow}\mathtt{II}]$
$\to_{\mathtt{ym}}$	$z'[z'{\leftarrow}\mathtt{I}][z{\leftarrow}y][x{\leftarrow}\mathtt{I}][y{\leftarrow}\mathtt{II}]$	$\to_{\mathtt{ye_{YN}}}$	$\mathtt{I}[z'{\leftarrow}\mathtt{I}][z{\leftarrow}y][x{\leftarrow}\mathtt{I}][y{\leftarrow}\mathtt{II}]$

CbS extension:

		$\to_{\mathtt{ye_{AY}}}$	$\mathtt{I}[z'{\leftarrow}\mathtt{I}][z{\leftarrow}\mathtt{II}][x{\leftarrow}\mathtt{I}][y{\leftarrow}\mathtt{II}]$
$\to_{\mathtt{ym}}$	$\mathtt{I}[z'{\leftarrow}\mathtt{I}][z{\leftarrow}z'[z'{\leftarrow}\mathtt{I}]][x{\leftarrow}\mathtt{I}][y{\leftarrow}\mathtt{II}]$	$\to_{\mathtt{ye_{YN}}}$	$\mathtt{I}[z'{\leftarrow}\mathtt{I}][z{\leftarrow}\mathtt{I}[z'{\leftarrow}\mathtt{I}]][x{\leftarrow}\mathtt{I}][y{\leftarrow}\mathtt{II}]$
$\to_{\mathtt{ym}}$	$\mathtt{I}[z'{\leftarrow}\mathtt{I}][z{\leftarrow}\mathtt{I}[z'{\leftarrow}\mathtt{I}]][x{\leftarrow}\mathtt{I}][y{\leftarrow}z'[z'{\leftarrow}\mathtt{I}]]$	$\to_{\mathtt{ye_{YN}}}$	$\mathtt{I}[z'{\leftarrow}\mathtt{I}][z{\leftarrow}\mathtt{I}[z'{\leftarrow}\mathtt{I}]][x{\leftarrow}\mathtt{I}][y{\leftarrow}\mathtt{I}[z'{\leftarrow}\mathtt{I}]]$

If we consider CbN evaluation, evaluation stops earlier: it would reach a normal form with $\mathtt{I}[z'{\leftarrow}\mathtt{I}][z{\leftarrow}y][x{\leftarrow}\mathtt{I}][y{\leftarrow}\mathtt{II}]$, that is after 4 multiplicative and 5 exponential steps. CbS evaluates more than CbN, hence the rewriting sequence ends only after 6 multiplicative and 8 exponential steps.

We now show a tight derivation for the term $(\lambda y.yy)(\mathtt{II})$, which is indeed indexed by $(6,8)$, as per tight correctness and completeness (Th. 29 and the text after it). For compactness, we shorten $\mathtt{norm}$ as $\mathtt{n}$. Moreover, to fit the derivation in the page, we first give the main derivation, giving a name to some sub-derivations which are shown after the main one.

- Main derivation:

$$\dfrac{\pi \vdash^{(1,3)} \lambda y.yy : [[\mathtt{n}]\to\mathtt{n},\mathtt{n},\mathtt{n}]\to\mathtt{n} \qquad \dfrac{\rho \vdash^{(1,2)} \mathtt{II} : [\mathtt{n}]\to\mathtt{n} \qquad (\rho_{\mathtt{n}} \vdash^{(1,1)} \mathtt{II} : \mathtt{n}) \times 3}{\vdash^{(4,5)} \mathtt{II} : [[\mathtt{n}]\to\mathtt{n},\mathtt{n},\mathtt{n}] \uplus [\mathtt{n}]}\,\mathsf{many}}{\vdash^{(6,8)} (\lambda y.yy)(\mathtt{II}) : \mathtt{n}}\,@$$

- Auxiliary ones:

$$\pi := \dfrac{\dfrac{y : [[\mathtt{n}]\to\mathtt{n}] \vdash^{(0,1)} y : [\mathtt{n}]\to\mathtt{n} \qquad \dfrac{y : [\mathtt{n}] \vdash^{(0,1)} y : \mathtt{n} \qquad y : [\mathtt{n}] \vdash^{(0,1)} y : \mathtt{n}}{y : [\mathtt{n},\mathtt{n}] \vdash^{(0,2)} y : [\mathtt{n}] \uplus [\mathtt{n}]}\,\mathsf{many}}{y : [[\mathtt{n}]\to\mathtt{n},\mathtt{n},\mathtt{n}] \vdash^{(1,3)} yy : \mathtt{n}}\,@}{\vdash^{(1,3)} \lambda y.yy : [[\mathtt{n}]\to\mathtt{n},\mathtt{n},\mathtt{n}]\to\mathtt{n}}\,\lambda$$

$$\rho_{\mathtt{n}} := \dfrac{\pi_{\mathtt{n}} \vdash^{(0,1)} \mathtt{I} : [[\mathtt{n}]\to\mathtt{n}] \qquad \dfrac{\dfrac{}{\vdash^{(0,0)} \mathtt{I} : \mathtt{n}}\,\mathsf{ax}_\lambda \qquad \dfrac{}{\vdash^{(0,0)} \mathtt{I} : \mathtt{n}}\,\mathsf{ax}_\lambda}{\vdash^{(0,0)} \mathtt{I} : [\mathtt{n}] \uplus [\mathtt{n}]}\,\mathsf{many}}{\vdash^{(1,1)} \mathtt{II} : \mathtt{n}}\,@$$

$$\rho := \dfrac{\dfrac{\dfrac{}{z : [[\mathtt{n}]\to\mathtt{n}] \vdash^{(0,1)} z : [\mathtt{n}]\to\mathtt{n}}\,\mathsf{ax}}{\vdash^{(0,1)} \mathtt{I} : [[\mathtt{n}]\to\mathtt{n}]\to([\mathtt{n}]\to\mathtt{n})}\,\lambda \qquad \dfrac{\pi_{\mathtt{n}} \vdash^{(0,1)} \mathtt{I} : [\mathtt{n}]\to\mathtt{n} \qquad \dfrac{}{\vdash^{(0,0)} \mathtt{I} : \mathtt{n}}\,\mathsf{ax}_\lambda}{\vdash^{(0,1)} \mathtt{I} : [[\mathtt{n}]\to\mathtt{n}] \uplus [\mathtt{n}]}\,\mathsf{many}}{\vdash^{(1,2)} \mathtt{II} : [\mathtt{n}]\to\mathtt{n}}\,@$$

Maximality of the CbS Strategy. Similarly to how we proved uniform normalization for $\to_{\mathtt{w}}$, we can prove that on closed terms the CbS strategy does reach a weak normal form whenever one exists – the key point being that $\to_{\mathtt{y}}$ does not stop too soon. This fact proves that $\simeq_C^{\mathtt{silly}}$ can equivalently be defined using $\to_{\mathtt{y}}$, as mentioned in Sect. 7. Moreover, by exploiting tight correctness and some of the rewriting properties of Sect. 4, we prove that the CbS strategy is maximal.

▶ **Proposition 31.** *Let t be closed and $t \to_{\mathtt{w}}^{h} a_s$ with a_s a strict answer.*

1. CbS is normalizing*: $t \to_{\mathtt{y}}^{k} a_s$ for some $k \in \mathbb{N}$;*
2. CbS is maximal*: $h \leq k$.*

Proof.

1. By completeness (Th. 14), $t \to_{\mathtt{w}}^{h} a_s$ implies typability of t (a strict answer is in particular a weak normal form), which in turn, by tight correctness (Th. 29), implies $t \to_{\mathtt{y}}^{k} a'_s$ for some $k \in \mathbb{N}$. By confluence (Th. 7), $a_s = a'_s$.
2. By postponement of $\to_{\mathtt{wgcv}}$ (Prop. 3) applied to $t \to_{\mathtt{w}}^{h} a_s$, we obtain a sequence $d : t \to_{\mathtt{w}\neg\mathtt{gcv}}^{h_1} s \to_{\mathtt{wgcv}}^{h_2} a_s$ with $h_1 + h_2 \geq h$, for some s. For the strategy sequence $t \to_{\mathtt{y}}^{k} a_s$ given by Point 1, the *moreover* part of the postponement property gives us a sequence $e : t \to_{\mathtt{y}\neg\mathtt{gcv}}^{k_1} u \to_{\mathtt{ygcv}}^{k_2} a_s$ with, crucially, $k_1 + k_2 = k$, for some u. By completeness of the type system (Th. 14), we obtain a derivation $\vdash^{(m,e)} t : \mathtt{norm}$. By weak correctness (Th. 11), we obtain $h_1 \leq m + e$. By tight correctness (Th. 29), $k_1 = m + e$. Thus, $h_1 \leq k_1$.

Now, note that both s and u are $\to_{\mathtt{w}\neg\mathtt{gcv}}$-normal, because otherwise, by (iterated) strict commutation of $\to_{\mathtt{w}\neg\mathtt{gcv}}$ and $\to_{\mathtt{wgcv}}$ (Lemma 6), we obtain that a_s is not $\to_{\mathtt{w}}$-normal, against hypothesis. Since $\to_{\mathtt{w}\neg\mathtt{gcv}}$ is confluent (Th. 7), $s = u$. Since $\to_{\mathtt{wgcv}}$ is diamond (Lemma 5), $h_2 = k_2$. Then $k = k_1 + k_2 = k_1 + h_2 \geq h_1 + h_2 = h$. ◀

Why not the λ-Calculus? In the λ-calculus, it is hard to specify via evaluation contexts the idea behind the CbS strategy of *evaluating arguments only when they are no longer needed*. The difficulty is specific to weak evaluation. For instance, for $t := (\lambda y.\lambda x.yy)u$ the CbS strategy should evaluate u before substituting it for y, because once u ends up under λx it shall be unreachable by weak evaluation. For $s := (\lambda y.(\lambda x.yy)r)u$, instead, the CbS strategy should not evaluate u before substituting it, because weak evaluation will reach the two occurrences of y, and one obtains a longer reduction by substituting u before evaluating it. Note that one cannot decide what to do with u in t and s by checking if y occurs inside the abstraction, because y occurs under λy in both t and s.

Multi types naturally make the right choices for t (evaluating u before substituting it) and s (substituting u before evaluating it), so a strategy matching exactly the bounds given by multi types needs to do the same choices. The LSC allows one to bypass the difficulty, by first turning β-redexes into explicit substitutions, thus exposing y out of abstractions in s but not in t, and matching what is measured by multi types. In the λ-calculus, we have not found natural ways of capturing this aspect (be careful: the example illustrates the problem but solving the problem requires more than just handling the example).

10 Conclusions

We introduce the weak silly substitution calculus and the CbS strategy by mirroring the properties of CbNeed with respect to duplication and erasure. Then, we provide evidence of the good design of the framework via a rewriting study of the calculus and the strategy, and by mirroring the semantic analyses of CbNeed via multi types by Kesner [38] (qualitative) and Accattoli et al. [12] (quantitative).

Conceptually, the main results are the operational equivalence of CbS and CbV, mirroring the one between CbN and CbNeed, and the exact measuring of CbS evaluation lengths via multi types, having the interesting corollary that CbS is *maximal* in the Weak SSC. It would be interesting to show that, dually, the CbNeed strategy computes the shortest reduction in the CbNeed LSC. We are not aware of any such result.

Our work also shows that CbV contextual equivalence $\simeq_C^{\mathtt{value}}$ is completely blind to the efficiency of evaluation. We think that it is important to look for natural refinements of $\simeq_C^{\mathtt{value}}$ which are less blind, or, in the opposite direction, exploring what are the minimal extensions of CbV that make $\simeq_C^{\mathtt{value}}$ efficiency-sensitive.

We would also like to develop categorical and game semantics for both CbS and CbNeed, to pinpoint the principles behind the wiseness and the silliness of duplications and erasures.

References

1 Beniamino Accattoli. An abstract factorization theorem for explicit substitutions. In Ashish Tiwari, editor, *23rd International Conference on Rewriting Techniques and Applications (RTA'12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan*, volume 15 of *LIPIcs*, pages 6–21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.RTA.2012.6.

2 Beniamino Accattoli. Proof nets and the call-by-value λ-calculus. *Theor. Comput. Sci.*, 606:2–24, 2015. doi:10.1016/j.tcs.2015.08.006.

3 Beniamino Accattoli. Proof nets and the linear substitution calculus. In Bernd Fischer and Tarmo Uustalu, editors, *Theoretical Aspects of Computing - ICTAC 2018 - 15th International Colloquium, Stellenbosch, South Africa, October 16-19, 2018, Proceedings*, volume 11187 of *Lecture Notes in Computer Science*, pages 37–61. Springer, 2018. doi:10.1007/978-3-030-02508-3_3.

4 Beniamino Accattoli. Exponentials as substitutions and the cost of cut elimination in linear logic. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 49:1–49:15. ACM, 2022. doi:10.1145/3531130.3532445.

5 Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. Distilling abstract machines. In Johan Jeuring and Manuel M. T. Chakravarty, editors, *Proceedings of the 19th ACM SIGPLAN international conference on Functional programming, Gothenburg, Sweden, September 1-3, 2014*, pages 363–376. ACM, 2014. doi:10.1145/2628136.2628154.

6 Beniamino Accattoli and Bruno Barras. Environments and the complexity of abstract machines. In *Proceedings of the 19th International Symposium on Principles and Practice of Declarative Programming (PPDP 2017)*, pages 4–16, 2017. doi:10.1145/3131851.3131855.

7 Beniamino Accattoli and Bruno Barras. The negligible and yet subtle cost of pattern matching. In Bor-Yuh Evan Chang, editor, *Programming Languages and Systems - 15th Asian Symposium, APLAS 2017, Suzhou, China, November 27-29, 2017, Proceedings*, volume 10695 of *Lecture Notes in Computer Science*, pages 426–447. Springer, 2017. doi:10.1007/978-3-319-71237-6_21.

8 Beniamino Accattoli, Eduardo Bonelli, Delia Kesner, and Carlos Lombardi. A nonstandard standardization theorem. In Suresh Jagannathan and Peter Sewell, editors, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 659–670. ACM, 2014. doi:10.1145/2535838.2535886.

9 Beniamino Accattoli, Stéphane Graham-Lengrand, and Delia Kesner. Tight typings and split bounds, fully developed. *J. Funct. Program.*, 30:e14, 2020. doi:10.1017/S095679682000012X.

10 Beniamino Accattoli and Giulio Guerrieri. Open call-by-value. In Atsushi Igarashi, editor, *Programming Languages and Systems - 14th Asian Symposium, APLAS 2016, Hanoi, Vietnam, November 21-23, 2016, Proceedings*, volume 10017 of *Lecture Notes in Computer Science*, pages 206–226, 2016. doi:10.1007/978-3-319-47958-3_12.

11 Beniamino Accattoli and Giulio Guerrieri. The theory of call-by-value solvability. *Proc. ACM Program. Lang.*, 6(ICFP):855–885, 2022. doi:10.1145/3547652.

12 Beniamino Accattoli, Giulio Guerrieri, and Maico Leberle. Types by need. In *Proceedings of the 28th European Symposium on Programming (ESOP 2019)*, pages 410–439, 2019. doi:10.1007/978-3-030-17184-1_15.

13 Beniamino Accattoli and Delia Kesner. The structural *lambda*-calculus. In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 381–395. Springer, 2010. doi:10.1007/978-3-642-15205-4_30.

14 Beniamino Accattoli and Adrienne Lancelot. Mirroring call-by-need, or values acting silly, 2024. arXiv:2402.12078.

15 Beniamino Accattoli and Maico Leberle. Useful open call-by-need. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 4:1–4:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.4.

16 Beniamino Accattoli and Luca Paolini. Call-by-value solvability, revisited. In Tom Schrijvers and Peter Thiemann, editors, *Functional and Logic Programming - 11th International Symposium, FLOPS 2012, Kobe, Japan, May 23-25, 2012. Proceedings*, volume 7294 of *Lecture Notes in Computer Science*, pages 4–16. Springer, 2012. doi:10.1007/978-3-642-29822-6_4.

17 Zena M. Ariola, Paul Downen, Hugo Herbelin, Keiko Nakata, and Alexis Saurin. Classical call-by-need sequent calculi: The unity of semantic artifacts. In Tom Schrijvers and Peter Thiemann, editors, *Functional and Logic Programming - 11th International Symposium, FLOPS 2012, Kobe, Japan, May 23-25, 2012. Proceedings*, volume 7294 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 2012. doi:10.1007/978-3-642-29822-6_6.

18 Zena M. Ariola, Matthias Felleisen, John Maraist, Martin Odersky, and Philip Wadler. The call-by-need lambda calculus. In Ron K. Cytron and Peter Lee, editors, *Conference Record of POPL'95: 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Francisco, California, USA, January 23-25, 1995*, pages 233–246. ACM Press, 1995. doi:10.1145/199448.199507.

19 Zena M. Ariola, Hugo Herbelin, and Alexis Saurin. Classical call-by-need and duality. In C.-H. Luke Ong, editor, *Typed Lambda Calculi and Applications - 10th International Conference, TLCA 2011, Novi Sad, Serbia, June 1-3, 2011. Proceedings*, volume 6690 of *Lecture Notes in Computer Science*, pages 27–44. Springer, 2011. doi:10.1007/978-3-642-21691-6_6.

20 Pablo Arrighi and Gilles Dowek. Lineal: A linear-algebraic lambda-calculus. *Log. Methods Comput. Sci.*, 13(1), 2017. doi:10.23638/LMCS-13(1:8)2017.

21 Thibaut Balabonski, Pablo Barenbaum, Eduardo Bonelli, and Delia Kesner. Foundations of strong call by need. *PACMPL*, 1(ICFP):20:1–20:29, 2017. doi:10.1145/3110264.

22 Thibaut Balabonski, Antoine Lanco, and Guillaume Melquiond. A strong call-by-need calculus. In Naoki Kobayashi, editor, *6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, July 17-24, 2021, Buenos Aires, Argentina (Virtual Conference)*, volume 195 of *LIPIcs*, pages 9:1–9:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSCD.2021.9.

23 Pablo Barenbaum, Eduardo Bonelli, and Kareem Mohamed. Pattern matching and fixed points: Resource types and strong call-by-need: Extended abstract. In David Sabel and Peter Thiemann, editors, *Proceedings of the 20th International Symposium on Principles and Practice of Declarative Programming, PPDP 2018, Frankfurt am Main, Germany, September 03-05, 2018*, pages 6:1–6:12. ACM, 2018. doi:10.1145/3236950.3236972.

24 Alexis Bernadet and Stéphane Lengrand. Complexity of strongly normalising λ-terms via non-idempotent intersection types. In *FOSSACS 2011*, pages 88–107, 2011. doi:10.1007/978-3-642-19805-2_7.

25 Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. A complete normal-form bisimilarity for state. In Mikolaj Bojanczyk and Alex Simpson, editors, *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11425 of *Lecture Notes in Computer Science*, pages 98–114. Springer, 2019. doi:10.1007/978-3-030-17127-8_6.

26 Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. Solvability = typability + inhabitation. *Log. Methods Comput. Sci.*, 17(1), 2021. URL: https://lmcs.episciences.org/7141.

27 Alberto Carraro and Giulio Guerrieri. A semantical and operational account of call-by-value solvability. In Anca Muscholl, editor, *Foundations of Software Science and Computation Structures - 17th International Conference, FOSSACS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, volume 8412 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 2014. doi:10.1007/978-3-642-54830-7_7.

28 Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In Martin Odersky and Philip Wadler, editors, *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000*, pages 233–243. ACM, 2000. doi:10.1145/351240.351262.

29 Ugo Dal Lago and Simone Martini. The weak lambda calculus as a reasonable machine. *Theor. Comput. Sci.*, 398(1-3):32–50, 2008. doi:10.1016/j.tcs.2008.01.044.

30 Daniel de Carvalho. *Sémantiques de la logique linéaire et temps de calcul*. Thèse de doctorat, Université Aix-Marseille II, 2007.

31 Daniel de Carvalho. Execution time of λ-terms via denotational semantics and intersection types. *Mathematical Structures in Computer Science*, 28(7):1169–1203, 2018. doi:10.1017/S0960129516000396.

32 Thomas Ehrhard. Collapsing non-idempotent intersection types. In Patrick Cégielski and Arnaud Durand, editors, *Computer Science Logic (CSL'12) - 26th International Workshop/21st Annual Conference of the EACSL, CSL 2012, September 3-6, 2012, Fontainebleau, France*, volume 16 of *LIPIcs*, pages 259–273. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.CSL.2012.259.

33 Thomas Ehrhard and Giulio Guerrieri. The bang calculus: an untyped lambda-calculus generalizing call-by-name and call-by-value. In James Cheney and Germán 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. doi:10.1145/2967973.2968608.

34 Claudia Faggian and Simona Ronchi Della Rocca. Lambda calculus and probabilistic computation. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785699.

35 Philippa Gardner. Discovering needed reductions using type theory. In Masami Hagiya and John C. Mitchell, editors, *Theoretical Aspects of Computer Software, International Conference TACS '94, Sendai, Japan, April 19-22, 1994, Proceedings*, volume 789 of *Lecture Notes in Computer Science*, pages 555–574. Springer, 1994. doi:10.1007/3-540-57887-0_115.

36 J.R. Hindley. *The Church-Rosser Property and a Result in Combinatory Logic*. PhD thesis, University of Newcastle-upon-Tyne, 1964.

37 Axel Kerinec, Giulio Manzonetto, and Simona Ronchi Della Rocca. Call-by-value, again! In Naoki Kobayashi, editor, *6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, July 17-24, 2021, Buenos Aires, Argentina (Virtual Conference)*, volume 195 of *LIPIcs*, pages 7:1–7:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSCD.2021.7.

38 Delia Kesner. Reasoning about call-by-need by means of types. In Bart Jacobs and Christof Löding, editors, *Foundations of Software Science and Computation Structures - 19th International Conference, FOSSACS 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 9634 of *Lecture Notes in Computer Science*, pages 424–441. Springer, 2016. doi:10.1007/978-3-662-49630-5_25.

39 Delia Kesner and Shane Ó Conchúir. Milner's lambda calculus with partial substitutions. Technical report, Paris 7 University, 2008. URL: http://www.pps.univ-paris-diderot.fr/~kesner/papers/shortpartial.pdf.

40 Delia Kesner, Loïc Peyrot, and Daniel Ventura. The spirit of node replication. In Stefan Kiefer and Christine Tasson, editors, *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings*, volume 12650 of *Lecture Notes in Computer Science*, pages 344–364. Springer, 2021. doi:10.1007/978-3-030-71995-1_18.

41 Delia Kesner, Alejandro Ríos, and Andrés Viso. Call-by-need, neededness and all that. In Christel Baier and Ugo Dal Lago, editors, *Foundations of Software Science and Computation Structures - 21st International Conference, FOSSACS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, volume 10803 of *Lecture Notes in Computer Science*, pages 241–257. Springer, 2018. doi:10.1007/978-3-319-89366-2_13.

42 Delia Kesner and Daniel Ventura. Quantitative types for the linear substitution calculus. In Josep Díaz, Ivan Lanese, and Davide Sangiorgi, editors, *Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*, volume 8705 of *Lecture Notes in Computer Science*, pages 296–310. Springer, 2014. doi:10.1007/978-3-662-44602-7_23.

43 A. J. Kfoury. A linearization of the lambda-calculus and consequences. *J. Log. Comput.*, 10(3):411–436, 2000. doi:10.1093/LOGCOM/10.3.411.

44 Olivier Laurent. *Étude de la polarisation en logique*. Thèse de doctorat, Université Aix-Marseille II, March 2002.

45 Paul Blain Levy. Call-by-push-value: Decomposing call-by-value and call-by-name. *High. Order Symb. Comput.*, 19(4):377–414, 2006. doi:10.1007/s10990-006-0480-6.

46 Giulio Manzonetto, Michele Pagani, and Simona Ronchi Della Rocca. New semantical insights into call-by-value λ-calculus. *Fundam. Informaticae*, 170(1-3):241–265, 2019. doi:10.3233/FI-2019-1862.

47 John Maraist, Martin Odersky, David N. Turner, and Philip Wadler. Call-by-name, call-by-value, call-by-need and the linear lambda calculus. *Theor. Comput. Sci.*, 228(1-2):175–210, 1999. doi:10.1016/S0304-3975(98)00358-2.

48 Robin Milner. Local bigraphs and confluence: Two conjectures: (extended abstract). In Roberto M. Amadio and Iain Phillips, editors, *Proceedings of the 13th International Workshop on Expressiveness in Concurrency, EXPRESS 2006, Bonn, Germany, August 26, 2006*, volume 175 of *Electronic Notes in Theoretical Computer Science*, pages 65–73. Elsevier, 2006. doi:10.1016/j.entcs.2006.07.035.

49 Gordon D. Plotkin. Call-by-name, call-by-value and the lambda-calculus. *Theor. Comput. Sci.*, 1(2):125–159, 1975. doi:10.1016/0304-3975(75)90017-1.

50 György E. Révész. A list-oriented extension of the lambda-calculus satisfying the Church-Rosser theorem. *Theor. Comput. Sci.*, 93(1):75–89, 1992. doi:10.1016/0304-3975(92)90212-X.

51 Alexis Saurin. On the relations between the syntactic theories of lambda-mu-calculi. In Michael Kaminski and Simone Martini, editors, *Computer Science Logic, 22nd International Workshop, CSL 2008, 17th Annual Conference of the EACSL, Bertinoro, Italy, September 16-19, 2008. Proceedings*, volume 5213 of *Lecture Notes in Computer Science*, pages 154–168. Springer, 2008. doi:10.1007/978-3-540-87531-4_13.

52 Kristian Støvring and Søren B. Lassen. A complete, co-inductive syntactic theory of sequential control and state. In Jens Palsberg, editor, *Semantics and Algebraic Specification, Essays Dedicated to Peter D. Mosses on the Occasion of His 60th Birthday*, volume 5700 of *Lecture Notes in Computer Science*, pages 329–375. Springer, 2009. doi:10.1007/978-3-642-04164-8_17.

53 Christopher P. Wadsworth. *Semantics and pragmatics of the lambda-calculus*. PhD Thesis, Oxford, 1971.