The Theory of Call-by-Value Solvability

BENIAMINO ACCATTOLI, Inria & LIX, École Polytechnique, UMR 7161, France
GIULIO GUERRIERI, Edinbugh Research Centre, Central Software Institute, Huawei, UK

The semantics of the untyped (call-by-name) λ-calculus is a well developed field built around the concept of solvable terms, which are elegantly characterized in many different ways. In particular, unsolvable terms provide a consistent notion of meaningless term. The semantics of the untyped *call-by-value* λ-calculus (CbV) is instead still in its infancy, because of some inherent difficulties but also because CbV solvable terms are less studied and understood than in call-by-name. On the one hand, we show that a carefully crafted presentation of CbV allows us to recover many of the properties that solvability has in call-by-name, in particular qualitative and quantitative characterizations via multi types. On the other hand, we stress that, in CbV, solvability plays a different role: identifying unsolvable terms as meaningless induces an inconsistent theory.

CCS Concepts: • **Theory of computation** → **Lambda calculus**; **Denotational semantics**; **Operational semantics**.

Additional Key Words and Phrases: solvability, call-by-value, intersection types.

ACM Reference Format:
Beniamino Accattoli and Giulio Guerrieri. 2022. The Theory of Call-by-Value Solvability. *Proc. ACM Program. Lang.* 6, ICFP, Article 121 (August 2022), 31 pages. https://doi.org/10.1145/3547652

1 INTRODUCTION

A semantics of the (untyped) λ-calculus can be simply seen as an equational theory over λ-terms. The λ-calculus is Turing-complete, thus there should be notions of terminating/defined/meaningful and diverging/undefined/meaningless computations corresponding to the ones of partial recursive functions. At the level of the equational theory, it is natural to have many different equivalence classes of meaningful terms, while one would expect to have a unique equivalence class of meaningless terms. That is, all meaningless terms should be equated, or *collapsed*.

Instinctively, one would identify being meaningful with *being (β-)normalizable* (a normal form being the result of computation), and thus, dually, being meaningless with being (β-)divergent. As it is often the case in the theory of λ-calculus, things are not as simple as that.

Theories that collapse all divergent terms, called here *normalizable* theories, have two related drawbacks. Firstly, the representation of partial recursive functions mapping the *everywhere undefined function* to the class of divergent terms is problematic, as it is not stable by composition. The crucial point is that such a notion of meaningless term is not stable by substitution. Secondly, and more importantly, normalizable theories are *inconsistent*, that is, because of the closure properties of theories, they end up equating *all* λ-terms. Therefore, there is a unique and *trivial* normalizable theory. We then say that divergent terms are not *collapsible*.

Authors' addresses: Beniamino Accattoli, Inria & LIX, École Polytechnique, UMR 7161, France, beniamino.accattoli@inria.fr; Giulio Guerrieri, Edinbugh Research Centre, Central Software Institute, Huawei, UK, giulio.guerrieri@huawei.com.

2475-1421/2022/8-ART121
https://doi.org/10.1145/3547652

One might suspect that being normalizable is not a meaningful predicate. But be careful: the issue is rather that it is too coarse to see all divergent terms as meaningless, that is, there is meaning also in some divergent terms. The point is that *normalizing terms are not the only meaningful ones*.

Solvability. These issues were first studied by Wadsworth [1971, 1976] and Barendregt [1971, 1977] in the '70s. They showed that both drawbacks of the normalizable theory disappear if meaningful/meaningless terms are rather identified with solvable/unsolvable terms. (Un)solvable terms can be equivalently defined in many ways. The usual definition is: *t is solvable if there exists a head context H sending t to the identity* $\mathsf{I} := \lambda z.z$, that is, such that $H\langle t\rangle \to_\beta^* \mathsf{I}$. The idea is that a solvable term t might be divergent but all its diverging subterms are removable via interactions with an environment that cannot simply discard t (enforced by the restriction to *head* contexts). As an example, a divergent term such as $x\Omega$ (with $\Omega := \delta\delta$ and $\delta := \lambda y.yy$) is solvable, because the head context $(\lambda x.\langle\cdot\rangle)\lambda y.\mathsf{I}$ sends it on the identity by erasing the diverging argument Ω. Consider instead Ω: no head contexts can erase its divergence and produce the identity, thus it is unsolvable. More generally, unsolvable terms are a strict subset of the diverging ones.

A compositional representation of partial recursive functions can then be given, as shown by Barendregt [1977], and the equational theory extending β-conversion with the collapse of all unsolvable terms—known as the *minimal sensible theory* $\mathcal{H}$—is *consistent*, that is, it does not equate all λ-terms. In particular, the consistent equational theories of important models of the λ-calculus such as Scott's D_∞ or the relational semantics do collapse all unsolvable terms.

Characterizations of Solvability. A natural question is whether the external ingredient in the definition of solvability, represented by the head context, can be somehow internalized. Wadsworth [1976] showed that it can: *a term t is solvable if and only if the head reduction of t terminates*. This is often referred to as the *operational characterization* of solvability. The characterization shows that, internally, *meaningful* should be associated to *head normalizable* rather than *normalizable*. Since *head normalizable* is a weaker predicate than *normalizable*, the solvable approach is *more meaningful*, in the sense that, as expected, it accepts more terms as meaningful. Additionally, the head normalizable predicate is a *refinement* of the normalizable one, as normalizable terms are *hereditarily head normalizable terms*, that is, terms that are head normalizable and the head arguments of which are hereditarily head normalizable.

Solvability can also be characterized via *intersection types* [Coppo and Dezani-Ciancaglini 1978, 1980], which are a theoretical notion of type mediating between semantic and operational properties: a term t is solvable if and only if t is typable with intersection types. Moreover, by adopting *non-idempotent* intersection types [de Carvalho 2007, 2018; Gardner 1994], also known as *multi types*, one can also extract quantitative operational information about solvable terms. Namely, the number of head reduction steps (which is a reasonable measure of time complexity for λ-terms, see [Accattoli and Dal Lago 2012]), and the size of the head normal form, as first shown by de Carvalho [2007, 2018]. Multi types are also relevant because the set of multi type judgments for a term t is a syntactic presentation of the relational semantics of t, a paradigmatic denotational model of the λ-calculus.

Semantics of Call-by-Value. Many variants of the λ-calculus have emerged. What is usually referred to as *the* λ-calculus could nowadays be more precisely referred to as the *(strong) call-by-name (CbN for short) λ-calculus*. Somewhat embarrassingly, it is the most studied of λ-calculi, and yet it is the one that it is *never* used in applications. Functional programming languages, in particular, often prefer Plotkin's [1975] *call-by-value* (*CbV* for short) λ-calculus, where β-redexes can fire only when the argument is a *value* (*i.e.*, not an application) and usually further restrict it to *weak reduction* (*i.e.*, out of abstractions) and to closed terms—what we shall refer to as *Closed CbV* (λ-calculus).

The denotational semantics of the CbV λ-calculus is less studied and understood than for CbN (some exceptions are [Egidi et al. 1992; Ehrhard 2012; Manzonetto et al. 2019; Pravato et al. 1999]). This is not by chance: as first shown by Paolini and Ronchi Della Rocca [2001; 1999; 2004], inherent complications arise when adapting semantic notions from CbN to CbV, basically for two reasons.

- *Difficulties with open terms*: while Closed CbV is an elegant setting, denotational semantics has to deal with *open* terms, and Plotkin's operational semantics is not adequate for that because of *premature* normal forms—see [Accattoli and Guerrieri 2016] for extensive discussions.
- *Inability to erase some divergent subterms*: while in CbN every term is erasable, in CbV only values are erasable. So, CbV solvable terms does not coincide with the CbN ones. In particular, the (CbN) solvable term $x\Omega$ given above is not solvable in CbV, because Ω cannot be erased.

The difficulty with open terms has the consequence that CbV solvability does not admit an *internal* operational characterization akin to the one for CbN [Wadsworth 1976], and thus it is not really an easily manageable notion. Additionally, some of the properties that solvable terms have in CbN are rather verified, in CbV, by another, larger set of terms, called here *scrutable terms*[1]. In particular, a term is typable with CbV intersection/multi types if and only if it is scrutable (instead of solvable).

Two Approaches to Call-by-Value Solvability. The literature has focused more on solvability than scrutability, exploring two opposite approaches towards the difficulties of studying it in CbV:

(1) *Disruptive*: replacing Plotkin's CbV calculus with another, extended CbV calculus so as to obtain a smoother framework, and in particular an easier theory of solvability;
(2) *Conservative*: considering Plotkin's CbV calculus as untouchable and striving harder to characterize semantic notions, and potentially build new ones.

One of the achievements of the disruptive approach is the operational characterization of both CbV scrutability and solvability due to Accattoli and Paolini [2012]. They introduce a CbV λ-calculus with let expressions which is isomorphic to the proof-net CbV representation of λ-calculus, called *value substitution calculus* (shortened to VSC), together with a *solving reduction* (called *stratified-weak* in [Accattoli and Paolini 2012]) and prove that (for possibly open terms):

- t is VSC-scrutable if and only if the weak reduction of t terminates;
- t is VSC-solvable if and only if the solving reduction of t terminates.

This is akin to Wadsworth's [1976] characterization in CbN via head reduction. In particular, the VSC characterizations show that if a term is VSC solvable then it is *hereditarily scrutable*, since solving reduction is defined by iterating weak reduction (under head abstractions).

The conservative approach is explored by García-Pérez and Nogueira [2016]. Inspired by a fine analysis CbN solvability, they strive to adapt some of its properties to CbV. They propose alternative notions of CbV solvability and scrutability (here referred to as by adding the prefix *GPN*)[2] that are not equivalent to the usual ones in Plotkin's CbV λ-calculus. Because of the difficulties of Plotkin's framework, their results are strictly weaker than in CbN and considerably more complex. They also lack denotational or type-theoretic justifications, *e.g.* via intersection types.

García-Pérez and Nogueira are also the first ones to clearly mention that—surprisingly—equational theories collapsing all CbV unsolvable terms are inconsistent[3]. Such a non-collapsibility is relevant because it shows that in CbV—in contrast to CbN—*unsolvable* does not mean *meaningless*, that is,

[1] Introduced by Paolini and Ronchi Della Rocca [1999], scrutable terms are those terms for which there is a (certain kind of) head context sending them to a *value* (rather than the identity as in solvability). They are called *potentially valuable* in the literature, but we prefer to use a lighter terminology. Inscrutable terms are also called *unsolvable of order 0* in the literature. Paolini [2001] used inscrutable (closed) terms to represent undefined partial recursive functions in Plotkin's CbV λ-calculus.
[2] We call *GPN-inscrutable terms* what they call *GPN-unsolvable terms of order 0*, which they prove to be collapsible.
[3] For this result, they point to [Paolini and Ronchi Della Rocca 1999], where it is mentioned but it is not stated nor proved. It follows instead from results in [Egidi et al. 1992], where however it is not stated nor mentioned.

there is meaning to be found in some CbV unsolvable terms. Such an essential point seems to have been neglected instead by the disruptive approach.

Closing the Schism. A reconciliation of the disruptive and the conservative approaches is obtained by Guerrieri et al. [2015, 2017]. On the one hand, they embrace the disruptive approach, as they study the *shuffling calculus* [Carraro and Guerrieri 2014], another extension of Plotkin's calculus which can be seen as a variant over the VSC (where let expressions are replaced by commuting conversions) and where Accattoli and Paolini's operational characterization of solvability smoothly transfers. On the other hand, they prove that *a λ-term t is solvable in the shuffling calculus if and only if it is solvable in Plotkin's CbV λ-calculus*. Therefore, the disruptive extension becomes a way to study the conservative notion of solvability for Plotkin's calculus.

Open Questions about CbV Solvability. These works paved the way for a theory of CbV solvability analogous to the one in CbN. Such a theory however is still lacking.

Semantically, the literature has somewhat neglected the important issue of collapsibility in CbV. Operationally, the proofs of equivalence of various definitions of CbN solvability do not carry over Plotkin's calculus, as pointed out by García-Pérez and Nogueira [2016].

Further delicate points concern the characterization of CbV solvable terms via type systems. In CbV, there are characterizations of solvable terms via intersection and multi types [Kerinec et al. 2021; Paolini and Ronchi Della Rocca 1999]. Those type systems, however, are *defective*: despite what is stated in [Kerinec et al. 2021; Paolini and Ronchi Della Rocca 1999], they do not verify subject reduction (and for [Paolini and Ronchi Della Rocca 1999] subject expansion fails as well), as we detail in [Accattoli and Guerrieri 2022, Appendix A]. Carraro and Guerrieri [2014] characterize CbV solvability using relational semantics, but their characterization is not purely semantic (or type-theoretic) because it also needs the syntactic notion of *CbV Taylor expansion* [Ehrhard 2012]. Additionally, from none of these characterizations it is possible to extract quantitative operational information. They all rely, indeed, on the shuffling calculus, for which it is unclear how to extract (from type derivations) the number of commuting conversion steps, and the time cost model of which is also unclear, see [Accattoli and Guerrieri 2016]. Accattoli and Guerrieri [2018] provide a quantitative characterization of CbV scrutability via multi types, but not of CbV solvability.

Contributions. In this paper we study all these questions, providing also a *quantitative* analysis of solvability via intersection types. Because of the quantitative aspect, we study solvability *in the VSC* rather than in the shuffling calculus. The VSC is indeed a better fit than the shuffling calculus for quantitative analyses, because its number of β steps *is* a reasonable time cost model and can be extracted from multi type derivations, as shown by Accattoli et al. [2021a,d].

The paper is divided in *two* parts. The first part deals with providing evidence for the robustness of our approach and clarifying some key aspects in the literature. Our contributions are:

(1) *Robustness: solvabilities coincide.* Following [Guerrieri et al. 2015, 2017], we prove that both solvability and contextual equivalence in the VSC coincide with the corresponding notions in Plotkin's calculus. Thus, similarly to the shuffling calculus, also the VSC is a disruptive tool which can be used to study CbV notions in a conservative way.
(2) *Operationally: alternative definitions of solvability.* In the VSC, we show how to catch the various equivalent definitions of solvability holding in CbN. And we also give a new equivalent definition that captures CbV solvability at the *open* level, instead than at the *strong* one.
(3) *Semantically: CbV collapsibility.* We point out that CbV inscrutable terms are collapsible and show why CbV unsolvable terms instead are not. Showing this crucial facts simply amounts to collect results already in the literature but which were never presented in this way, and validates the *meaningful-as-scrutable* approach in CbV, instead of *meaningful-as-solvable*.

In the second part, we provide an in-depth study of the relationship between CbV solvability and multi types. The contributions are:

(1) *Multi types and CbV solvability*: we characterize CbV solvability using CbV *multi types* [Ehrhard 2012], which are strongly related to linear logic. Namely, we prove that a term is CbV solvable if and only if it is typable with a certain kind of multi types deemed *solvable* and inspired by Paolini and Ronchi Della Rocca [1999];
(2) *Bounds from types*: refining our solvable types, we extract the number of steps of the solving reduction on a solvable term, together with the size of the solving normal form. This study re-casts de Carvalho's results in a CbV setting, but it is more than a simple adaptation, as the CbV case requires new concepts.

The contributions of the second part of the paper are the most elaborate. The beginning of Section 7 provides an introduction to multi types, references to the literature, and an overview of our results.

The Big Picture. While solvability is certainly subtler in CbV than in CbN, our contributions show that, if the presentation of CbV is carefully crafted, then a solid theory of CbV solvability is possible. In fact, we obtain a theory comparable to the one in CbN.

Because of the duality between CbN and CbV in classical logic, the literature tends to see these two settings as mirror images of each other. While we show that solvability can indeed be defined and characterized in similar ways in CbN and CbV, we also find that it has inherently different roles for the semantics of the two settings, because of the non-collapsibility of CbV unsolvable terms. It might look as a negative result, but it actually sheds a positive light on CbV. It shows indeed that the semantic theory of CbV is *strictly finer* than the CbN one, as unsolvability is too coarse for capturing meaningless terms in CbV, where the right approach is the finer one of inscrutability.

It is important to stress that the non-collapsibility of CbV unsolvable terms does not mean that CbV solvability is uninteresting, similarly to how the inconsistency of the normalizable theory does not mean that normalization is uninteresting.

Methodology. The paper is built around a methodology which in our opinion is a further contribution to the theory of CbV. There are two correlated points:

(1) *Irrelevance*: we use the VSC as a core calculus, which is sufficient for computing results and characterizing solvability. We also introduce the concept of *irrelevant* extension or subtraction. The idea is that there are some rules and equivalences that: (a) can be added or removed without breaking termination and confluence, and (b) can be postponed.
As extensions, we consider a structural equivalence and in [Accattoli and Guerrieri 2022, Appendix B] we discuss a rule related to Moggi's [1988; 1989] computational λ-calculus. As subtractions, we consider a sub-relation of the core VSC—the substitution of variables—inspired by work on the study of cost models for CbV [Accattoli et al. 2019a; 2021a; 2016; 2018; 2015; 2017]. The VSC without the substitution of variables is a complete operational *sub-core*. Actually, such a sub-core has some operational properties *not* available in the VSC, and plays a role in the proof of some properties of CbV inscrutable and unsolvable terms.
(2) *Normal forms*: additionally, the sub-core admits a neat inductive description of CbV normal forms in terms of *inert terms* and *fireballs*, akin to the one for CbN and used throughout the paper. The role of inert terms, in particular, is crucial in the study of multi types, and it is also used to give a new alternative definition of CbV solvability.

These points are key technical differences between our study of CbV solvability and the other ones in the literature [Accattoli and Paolini 2012; Carraro and Guerrieri 2014; García-Pérez and Nogueira 2016; Guerrieri et al. 2015, 2017; Kerinec et al. 2021; Paolini and Ronchi Della Rocca 1999]. They seem minor details but they can also be understood from more conceptual points of view.

Firstly, CbV is a modular setting organized in *two* levels: there is a core which can be safely extended with further rewriting rules, enriching the equational theory and the flexibility of the calculus. A similar point of view is also advocated by Manzonetto et al. [2019]. Our core, however, is smaller, as their core (that is, the shuffling calculus) contains part of our structural equivalence.

Secondly, according to the disruptive approach to CbV, the problem with Plotkin's calculus for CbV is about premature normal forms, and one needs to extend such a calculus in order to solve it. This is undeniable, and already studied at length, see [Accattoli and Guerrieri 2016] for an overview. There is however a second essential ingredient for obtaining a good semantic theory, the importance of which—we believe—has not been stressed enough so far: *having a neat inductive description of normal forms*. The various extensions of Plotkin's calculus do not necessarily have neat grammars for normal forms. Our contribution here is to show both the relevance of neat normal forms and the fact that they are connected to the (non-)substitution of variables.

Further Related Work. Another framework where solvability is subtler than in CbN is CbN extended with pattern matching, as shown by Bucciarelli et al. [2021]. The literature contains many CbV calculi extending Plotkin's, for instance [Curien and Herbelin 2000; Dyckhoff and Lengrand 2007; Espírito Santo 2020; Herbelin and Zimmermann 2009; Maraist et al. 1999; Moggi 1988, 1989; Sabry and Felleisen 1993; Sabry and Wadler 1997]. Fireballs and inert terms (under other names) were first considered by Paolini and Ronchi Della Rocca [1999; 2004], and then by Grégoire and Leroy [2002]. Their importance, however, has been recognized by the study of cost models for CbV.

Proofs. Omitted proofs and other details are in the Appendix of [Accattoli and Guerrieri 2022], the long version of this paper.

2 PRELIMINARIES AND NOTATIONS IN REWRITING

This technical section recalls some well-known notions and facts in rewrite theory, and introduces some notations used in the rest of the paper. We suggest skimming over them on the first reading.

For a binary relation $\to_{\mathsf{r}}$ on a set of terms, $\to_{\mathsf{r}}^*$ is its reflexive-transitive closure, $\to_{\mathsf{r}}^+$ is its transitive closure, $=_{\mathsf{r}}$ is its symmetric, transitive, reflexive closure. The *transpose* of $\to_{\mathsf{r}}$ is denoted by ${}_{\mathsf{r}}\!\leftarrow$.

Given a binary relation $\to_{\mathsf{r}}$, an r-*reduction sequence* (or simply *reduction sequence* if unambiguous) is a finite sequence of terms $d = (t_i)_{0\le i\le n}$ (for some $n \ge 0$) such that $t_i \to_{\mathsf{r}} t_{i+1}$ for all $1 \le i < n$; we write $d\colon t \to_{\mathsf{r}}^* u$ if $t_0 = t$ and $t_n = u$, and we then say that t r-*reduces* to u. The *length* n of d is denoted by $|d|$, and $|d|_a$ is the number of *a-steps* (*i.e.* the number of $t_i \to_a t_{i+1}$ for some $1 \le i < n$) in d, for a given sub-relation $\to_a \subseteq \to_{\mathsf{r}}$. We write $t \to_{\mathsf{r}}^k u$ if there exists $d\colon t \to_{\mathsf{r}}^* u$ with $|d| = k \ge 0$.

A term t is r-*normal* if there is no u such that $t \to_{\mathsf{r}} u$. A reduction sequence $d\colon t \to_{\mathsf{r}}^* u$ is r-*normalizing* if u is r-normal. A term t is *(weakly)* r-*normalizing* if there is a r-normalizing reduction sequence $d\colon t \to_{\mathsf{r}}^* u$; and t is *strongly* r-*normalizing* if there is no *diverging reduction sequence* from t, that is, there is no infinite sequence $(t_i)_{i\in\mathbb{N}}$ such that $t_0 = t$ and $t_i \to_{\mathsf{r}} t_{i+1}$ for all $i \in \mathbb{N}$ (in this case we also say that $\to_{\mathsf{r}}$ is *terminating* on t). Clearly, strong r-normalization implies weak r-normalization. A relation $\to_{\mathsf{r}}$ is *strongly normalizing* if every term t is strongly r-normalizing.

A relation $\to_{\mathsf{r}}$ is *confluent* if $u_1\ {}_{\mathsf{r}}^*\!\leftarrow t \to_{\mathsf{r}}^* u_2$ implies $u_1 \to_{\mathsf{r}}^*\ s\ {}_{\mathsf{r}}^*\!\leftarrow u_2$ for some s. It is well-known that if $\to_{\mathsf{r}}$ is confluent then:

(1) *Uniqueness of the normal form:* any term t has at most one normal form (*i.e.* if $t \to_{\mathsf{r}}^* u$ and $t \to_{\mathsf{r}}^* s$ with u and s r-normal, then $u = s$);
(2) *Church-Rosser:* for every terms t and u, if $t =_{\mathsf{r}} u$ then $t \to_{\mathsf{r}}^*\ s\ {}_{\mathsf{r}}^*\!\leftarrow u$ for some s.

A relation $\to_{\mathsf{r}}$ is *diamond* if $u_1\ {}_{\mathsf{r}}\!\leftarrow t \to_{\mathsf{r}} u_2$ and $u_1 \neq u_2$ imply $u_1 \to_{\mathsf{r}}\ s\ {}_{\mathsf{r}}\!\leftarrow u_2$ for some s. It is well-known that if $\to_{\mathsf{r}}$ is diamond then:

(1) *Confluence:* $\to_{\mathsf{r}}$ is confluent;

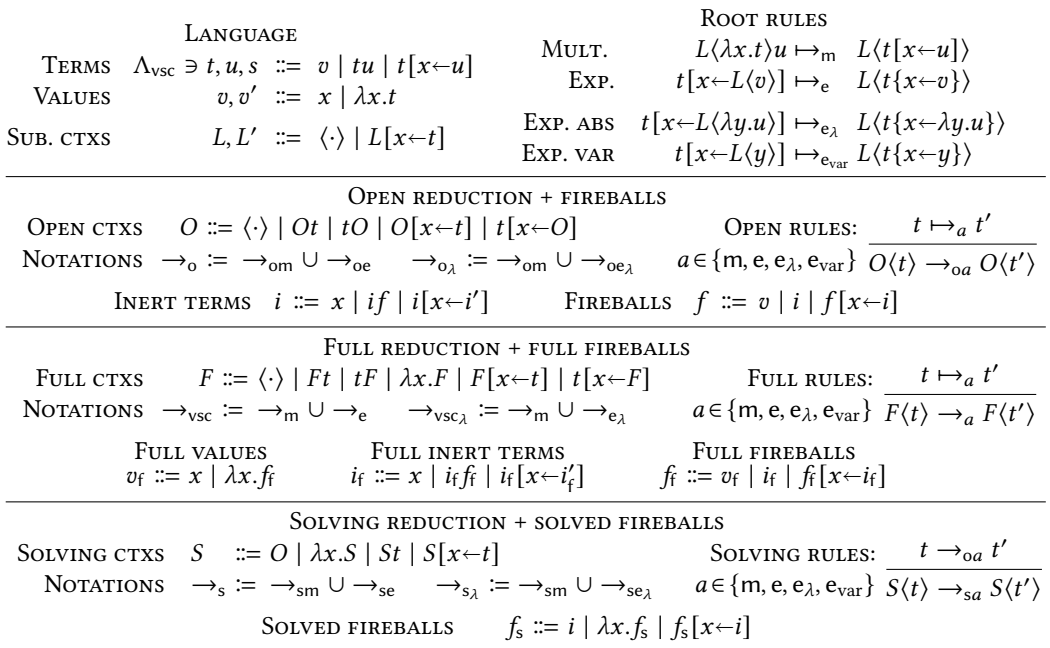

Fig. 1. Value Substitution Calculus and its 3 context closures and fireballs (open, full, solving).

(2) *Random descent:* all r-reduction sequences with the same start and end terms have the same length (*i.e.* if $d\colon t \rightarrow_r^* u$ and $d'\colon t \rightarrow_r^* u$ then $|d| = |d'|$);
(3) *Uniformity:* for any term t, t is weakly r-normalizing if and only if t is strongly r-normalizing.

Two relations $\rightarrow_{r_1}$ and $\rightarrow_{r_2}$ *strongly commute* if $u_1\ {}_{r_1}\!\leftarrow t \rightarrow_{r_2} u_2$ implies $u_1 \rightarrow_{r_2} s\ {}_{r_2}\!\leftarrow u_2$ for some s. If $\rightarrow_{r_1}$ and $\rightarrow_{r_2}$ strongly commute and are diamond, then

(1) *Diamond of the union:* reduction $\rightarrow_r := \rightarrow_{r_1} \cup \rightarrow_{r_2}$ is diamond;
(2) *Random descent bis:* all r-reduction sequences with the same start and end terms have the same number of any kind of steps (if $d\colon t \rightarrow_r^* u$ and $e\colon t \rightarrow_r^* u$ then $|d|_{r_1} = |e|_{r_1}$ and $|d|_{r_2} = |e|_{r_2}$).

3 VALUE SUBSTITUTION CALCULUS

In this section we define the *value substitution calculus* (shortened to VSC) [Accattoli and Paolini 2012]. Intuitively, the VSC is a CbV λ-calculus extended with let-expressions, as is common for CbV λ-calculi such as Moggi's [1988; 1989] one. We do however replace a let-expression let $x = u$ in t with a more compact *explicit substitution* (ES for short) notation $t[x\leftarrow u]$, which binds x in t. Moreover, our let/ES does not fix an order of evaluation between t and u, in contrast to many papers in the literature (*e.g.* Levy et al. [2003]; Sabry and Wadler [1997]) where u is evaluated first.

The reduction rules of VSC are slightly unusual as they use *contexts* both to allow one to reduce redexes located in subterms, which is standard, *and* to define the redexes themselves, which is less standard—these kind of rules is called *at a distance*. The rewriting rules in fact mimic exactly cut-elimination on proof nets, via Girard's [1987] CbV translation $(A \Rightarrow B)^v = !(A^v \multimap B^v)$ of intuitionistic logic into linear logic, see [Accattoli 2015]. We endow the terms of VSC with various notions of reductions. All the definitions are in Figure 1, the next paragraphs explain them in order.

Terms and Contexts. Terms may be *applications* tu, *values* (*i.e.* variables $x, y, z, \ldots$, and *abstractions* $\lambda x.t$) and *explicit substitutions* $t[x\leftarrow u]$. The set of *free* (resp. *bound*) *variables* of a term t, denoted by

$\mathrm{fv}(t)$ (resp. $\mathrm{bv}(t)$), is defined as expected, abstractions and ES being the only binding constructors. Terms are identified up to α-renaming of bound variables. We use $t\{x{\leftarrow}u\}$ for the capture-avoiding substitution of t for each free occurrence of x in t.

All along the paper we use (many notions of) *contexts, i.e.* terms with exactly one hole, noted $\langle\cdot\rangle$. Plugging a term t in a context C, noted $C\langle t\rangle$, possibly captures free variables of t. For instance $(\lambda x.\langle\cdot\rangle)\langle x\rangle = \lambda x.x$, while $(\lambda x.y)\{y{\leftarrow}x\} = \lambda z.x$. Figure 1 defines the notions of context that we use.

Root rewriting rules. In VSC, there are two main rewrite rules, the *multiplicative* one $\to_{\mathsf{m}}$ and the *exponential* one $\to_{\mathsf{e}}$ (the terminology comes from the connection between VSC and linear logic), and both work *at a distance*: they use contexts even in the definition of their *root* rules (that is, before the contextual closure). Their definition is based on *substitution contexts* L, which are lists of ES. In Figure 1, the root rule $\mapsto_{\mathsf{m}}$ (resp. $\mapsto_{\mathsf{e}}$) is assumed to be capture-free, so no free variable of u (resp. t) is captured by the substitution context L (by possibly α-renaming on-the-fly).

Examples: $(\lambda x.y)[y{\leftarrow}t]u \mapsto_{\mathsf{m}} y[x{\leftarrow}u][y{\leftarrow}t]$ and $(\lambda z.xx)[x{\leftarrow}y[y{\leftarrow}t]] \mapsto_{\mathsf{e}} (\lambda z.yy)[y{\leftarrow}t]$. An example with on-the-fly α-renaming is $(\lambda x.y)[y{\leftarrow}t]y \mapsto_{\mathsf{m}} z[x{\leftarrow}y][z{\leftarrow}t]$.

A key point is that β-redexes are decomposed via ES: the *by-value* restriction is on ES-redexes, *not* on β-redexes, because only values can be substituted. The multiplicative rule $\mapsto_{\mathsf{m}}$ fires a β-redex at a distance and generates an ES even when the argument is not a value. The CbV discipline is entirely encoded in the exponential rule $\to_{\mathsf{e}}$ (see Figure 1): it can fire an ES performing a substitution only when its argument is a *value* (*i.e.* a variable or an abstraction) up to a list of ES. This means that only values can be duplicated or erased. It is useful to split the exponential root rule $\mapsto_{\mathsf{e}}$ in two disjoint rules, depending on whether it is an abstraction (rule $\mapsto_{\mathsf{e}_\lambda}$) or a variable ($\mapsto_{\mathsf{e}_{\mathsf{var}}}$) that it is substituted.

We shall consider 3 different contextual closures for the given rules. For all of them, rule $\mapsto_{\mathsf{e}_{\mathsf{var}}}$ shall be postponable without altering the properties of the calculus (Prop. 3.7). Actually, the reductions without $\mapsto_{\mathsf{e}_{\mathsf{var}}}$ shall have stronger properties, crucial for some of our results.

With respect to the explanations in the introduction, our *core* calculus is the VSC with its three contextual closures. The irrelevant extension shall be considered in Sect. 4. The *sub-core* is instead obtained by removing $\mapsto_{\mathsf{e}_{\mathsf{var}}}$ from the three contextual closures, and it is justified in Sect. 3.4.

3.1 The Open VSC

The first contextual closure is the *open* one, where rewriting is forbidden under abstraction and terms are possibly open (but not necessarily). It is obtained via (possibly) open contexts O (see Figure 1). We consider both the reduction that substitutes variables, noted $\to_{\mathsf{o}}$, and the one that does not, noted $\to_{\mathsf{o}_\lambda}$ (indeed, note that $\to_{\mathsf{o}_\lambda} = \to_{\mathsf{o}} \setminus \to_{\mathsf{e}_{\mathsf{var}}}$). Examples:

$$t[x{\leftarrow}(\lambda y.u)[z{\leftarrow}s]q] \to_{\mathsf{om}} t[x{\leftarrow}u[y{\leftarrow}q][z{\leftarrow}s]] \qquad t((xx)[x{\leftarrow}y[z{\leftarrow}u]]) \to_{\mathsf{oe}_{\mathsf{var}}} t((yy)[z{\leftarrow}u])$$
$$((xx)[x{\leftarrow}\lambda y.z]t)[w{\leftarrow}u] \to_{\mathsf{oe}_\lambda} ((\lambda y.z)(\lambda y.z)t)[w{\leftarrow}u] \qquad \lambda z.((xx)[x{\leftarrow}\lambda y.z]) \not\to_{\mathsf{oe}_\lambda} \lambda z.((\lambda y.z)\lambda y.z)$$

Normal forms for $\to_{\mathsf{o}_\lambda}$ admit a neat inductive description via *inert terms* and *fireballs*.

Inert Terms and Fireballs. CbV is about *values*, and, if terms are closed, normal forms are abstractions. In going beyond the closed setting, a finer view is required. First, the notion of normal form in the Open VSC is more generally given by the mutually defined notions of *inert terms* and *fireballs* in Figure 1. Second, variables are *both* values and inert terms. This is on purpose, because they have the properties of both kinds of term.

Examples: $\lambda x.y$ is a fireball as an abstraction, while $y(\lambda x.x)$, xy, and $(z(\lambda x.x))(zz)(\lambda y.(zy))$ are fireballs as inert terms. The grammars also allow to have ES containing inert terms around abstractions and applications: $(\lambda x.y)[y{\leftarrow}zz]$ is a fireball and $x[x{\leftarrow}y(\lambda x.x)]y$ is an inert term. One of the key points of inert terms is that they have a *free* head variable (in particular they are open). In [Grégoire and Leroy 2002], inert terms are called *accumulators*, and fireballs are called *values*.

Normal forms for $\to_{\mathsf{o}_\lambda}$ are exactly fireballs. Note that $x[x{\leftarrow}y]$ is an inert term and it is not $\to_{\mathsf{oe}_{\mathsf{var}}}$ normal, thus not $\to_{\mathsf{o}}$ normal. Normal forms for $\to_{\mathsf{o}}$ are a slightly stricter subset of fireballs (they are fireballs without ES of shape $[x{\leftarrow}L\langle y\rangle]$), with a similar but less neat and omitted inductive description. We shall show that $\to_{\mathsf{oe}_{\mathsf{var}}}$ is postponable and strongly normalizing, allowing us to take fireballs as our reference notion of open normal form. The same approach shall be followed for the other contextual closures.

In the literature, fireballs have been considered for different open CbV calculi (the VSC and the fireball calculus), and their definition depends on the calculus. They are however characterized by the same operational property (fireballs are the normal forms for the open reduction of the chosen calculus) and they correspond to each other, see Accattoli and Guerrieri [2016]. The name *fireball*, due to Accattoli and Sacerdoti Coen [2015], is a pun: in the fireball calculus, a β-redex can be *fired* only when the argument is a fireball, so fireballs are the *fireable* terms, more catchily called *fireballs*.

Proposition 3.1 (Basic properties of open reduction).

(1) Strong commutation: *reductions* $\to_{\mathsf{om}}$, $\to_{\mathsf{oe}_\lambda}$, *and* $\to_{\mathsf{oe}_{\mathsf{var}}}$ *are pairwise strongly commuting.*
(2) Diamond: *reductions* $\to_{\mathsf{o}}$ *and* $\to_{\mathsf{o}_\lambda}$ *are diamond (separately).*
(3) Normal forms: *t is* o_λ*-normal if and only if t is a fireball. If t is* o*-normal then it is a fireball.*

Diamond of $\to_{\mathsf{o}}$ and strong commutation of $\to_{\mathsf{om}}$ and $\to_{\mathsf{oe}}$ are technical facts (see Section 2 for definitions) with relevant consequences: $\to_{\mathsf{o}}$ is confluent and its non-determinism is only apparent, because if an o-reduction sequence from t reaches a o-normal form u, then *every* o-sequence from t eventually ends in u; and all these sequences have the *same length* and *same number* of m-steps and e-steps. This is essential for measuring them via multi types in the second part of the paper. The same properties shall hold for solving reduction.

3.2 The Strong/Full VSC

To avoid notation clashes between the solving and strong reductions (both would start with 's'), we refer to the *strong* one as to the *full* one. The Full VSC allows rewrite rules to fire everywhere in a term, via full contexts F (see Figure 1). Note that $\to_{\mathsf{vsc}_\lambda} = \to_{\mathsf{vsc}} \setminus \to_{\mathsf{e}_{\mathsf{var}}}$, *i.e.*, $\to_{\mathsf{vsc}_\lambda}$ is the full reduction that does not substitute variables. Full fireballs are obtained by iterating the fireball construction under all abstractions. Full reductions $\to_{\mathsf{vsc}}$ and $\to_{\mathsf{vsc}_\lambda}$ are confluent but not diamond, just consider the example below ($\mathsf{I} := \lambda z.z$):

$$\begin{array}{ccccc} (xx)[x{\leftarrow}\lambda y.\mathsf{II}] & & \to_{\mathsf{m}} & & (xx)[x{\leftarrow}\lambda y.z[z{\leftarrow}\mathsf{I}]] \\ \downarrow_{\mathsf{e}_\lambda} & & & & \downarrow_{\mathsf{e}_\lambda} \\ (\lambda y.\mathsf{II})(\lambda y.\mathsf{II}) & \to_{\mathsf{m}} & (\lambda y.z[z{\leftarrow}\mathsf{I}])(\lambda y.\mathsf{II}) & \to_{\mathsf{m}} & (\lambda y.z[z{\leftarrow}\mathsf{I}])(\lambda y.z[z{\leftarrow}\mathsf{I}]) \end{array}$$

Proposition 3.2 (Basic properties of the full reduction).

(1) Confluence [Accattoli and Paolini 2012]: *reductions* $\to_{\mathsf{vsc}}$ *and* $\to_{\mathsf{vsc}_\lambda}$ *are confluent.*
(2) Normal forms: *a term is* vsc_λ*-normal if and only if it is a full fireball. If a term is* vsc*-normal then it is a full fireball.*

In the Full VSC, we shall also use the *valuability property* of $\to_{\mathsf{o}}$, *i.e.* that $\to_{\mathsf{o}}$ is enough to reach a value (Prop. 3.3.1 below). It is slightly weaker than a normalization theorem, because it concerns a specific kind of open normal form, values, and not all open normal forms, as it leaves out inert terms. A more general normalization property also hold, given as Point 2 of the next proposition, and that shall be proved in Section 10 using type theoretic means. Point 3 follows from Point 2 and the irrelevance of $\to_{\mathsf{e}_{\mathsf{var}}}$, which shall be introduced in Section 3.5.

Proposition 3.3 (Properties of open reduction with respect to full reduction).

(1) Valuability: *if $t \to^*_{\mathsf{vsc}} v$ for some value v, then $t \to^*_{\mathsf{o}} v'$ for some value v'.*

(2) Normalization: *if* $t \to_{\mathsf{vsc}}^* u$ *for some* o*-normal* u*, then* $t \to_{\mathsf{o}}^* s$ *for some* o*-normal* s.
(3) Normalization 2: *if* $t \to_{\mathsf{vsc}_\lambda}^* f$ *for some fireball* f*, then* $t \to_{\mathsf{o}_\lambda}^* f'$ *for some fireball* f'.

3.3 Solving Reduction

Solving reduction $\to_{\mathsf{s}}$ [Accattoli and Paolini 2012] is in between the full and open ones: it restricts $\to_{\mathsf{vsc}}$ but extends $\to_{\mathsf{o}}$. It iterates open reduction under head abstractions *only*, via the notion of *solving context* S (see Figure 1). For instance, the extension under head abstractions gives $\lambda x.\mathsf{I}\mathsf{I} \to_{\mathsf{sm}} \lambda x.(z[z{\leftarrow}\mathsf{I}]) \to_{\mathsf{se}} \lambda x.\mathsf{I}$. But reduction under non-head abstractions is forbidden: $y(\lambda x.\mathsf{I}\mathsf{I}) \not\to_{\mathsf{sm}} y(\lambda x.(z[z{\leftarrow}\mathsf{I}]))$. Solved fireballs iterate the fireball structure under head abstractions only.

Proposition 3.4 (Properties of solving reduction).

(1) Strong commutation: *reductions* $\to_{\mathsf{sm}}$, $\to_{\mathsf{se}_\lambda}$, *and* $\to_{\mathsf{se}_{\mathsf{var}}}$ *are pairwise strongly commuting.*
(2) Diamond*: reductions* $\to_{\mathsf{s}}$ *and* $\to_{\mathsf{s}_\lambda}$ *are diamond (separately).*
(3) Normal forms*: a term is* s_λ*-normal if and only if it is a solved fireball. If a term is* s*-normal then it is a solved fireball.*

CbV solvability shall be introduced in Section 5. From that point on, solving reduction shall play a crucial role. In particular, we shall also use its following properties.

Proposition 3.5 (Further properties of solving reduction).

(1) Normalization: *if* $t \to_{\mathsf{vsc}}^* u$ *for some* s*-normal* u*, then* $t \to_{\mathsf{s}}^* s$ *for some* s*-normal* s.
(2) Normalization 2: *if* $t \to_{\mathsf{vsc}_\lambda}^* f_{\mathsf{s}}$ *with* f_{s} *solved fireball, then* $t \to_{\mathsf{s}_\lambda}^* f_{\mathsf{s}}'$ *for some solved fireball* f_{s}'.
(3) Stability by extraction from a head context: *if* $H\langle t\rangle \to_{\mathsf{s}}^* u$ *for some head context* H *and* s*-normal* u*, then* $t \to_{\mathsf{s}}^* s$ *for some* s*-normal* s.

3.4 A Digression: Time Cost Model

Here we recall the reasonable time cost model for the VSC. It is not used anywhere in the paper, but it is part of the motivations for adopting the VSC and pursuing a quantitative study via multi types.

External Reduction and Reasonable Time. Accattoli et al. [2021a] introduce the *external reduction* of VSC, a sub-reduction of $\to_{\mathsf{vsc}_\lambda}$ that is diamond, extends $\to_{\mathsf{s}_\lambda}$ and computes vsc-normal forms. They prove that the number of multiplicative steps of external reduction to vsc-normal form is a reasonable time cost model for full CbV. The exclusion of $\to_{\mathsf{e}_{\mathsf{var}}}$ is a detail and does not affect the result, as the postponement of $\to_{\mathsf{e}_{\mathsf{var}}}$ actually preserves the number of multiplicative steps $\to_{\mathsf{m}}$, as we shall see in forthcoming Section 3.5 via the notion of irrelevance.
Accattoli et al. [2021d] additionally prove that external reduction is *normalizing* (in the untyped calculus), that is, that it reaches the normal form whenever it exists, thus giving to external reduction the same status of leftmost-outermost evaluation in CbN.

Solving Reduction and Reasonable Time. Solving reduction $\to_{\mathsf{s}}$ is a strict sub-relation of external reduction, thus its number of multiplicative steps also is a reasonable time cost model (the same holds for $\to_{\mathsf{s}_\lambda}$, and for $\to_{\mathsf{o}}$ and $\to_{\mathsf{o}_\lambda}$).

3.5 Irrelevance of Variable Exponential Steps

In VSC some sub-reductions of $\to_{\mathsf{vsc}}$ can be neglected because they are computationally *irrelevant*, in that they can be postponed and do not jeopardize normalization. Typically, this is the case for $\to_{\mathsf{e}_{\mathsf{var}}}$ and its variants, but there shall also be other cases.

Definition 3.6 (Irrelevance). Let R, S be binary relations on Λ_{vsc}. We say R is *S-irrelevant* if for every $t, u \in \Lambda_{\mathsf{vsc}}$:

- *Postponement:* if $d : t\ (S \cup R)^* u$ then there is $d' : t\ S^* R^* u$ with $|d'|_{\mathsf{m}} = |d|_{\mathsf{m}}$; and
- *Termination:* S is weakly (resp. strongly) normalizing on t if and only if so is $S \cup R$.

Proposition 3.7 (Irrelevance of $\to_{\mathsf{e_{var}}}$, $\to_{\mathsf{oe_{var}}}$, and $\to_{\mathsf{se_{var}}}$). *Reduction $\to_{\mathsf{e_{var}}}$ is $\to_{\mathsf{vsc}_\lambda}$-irrelevant, reduction $\to_{\mathsf{oe_{var}}}$ is $\to_{\mathsf{o}_\lambda}$-irrelevant, and reduction $\to_{\mathsf{se_{var}}}$ is $\to_{\mathsf{s}_\lambda}$-irrelevant.*

Distinguishing between $\to_{\mathsf{vsc}}$ and $\to_{\mathsf{vsc}_\lambda}$ (or between $\to_{\mathsf{s}}$ and $\to_{\mathsf{s}_\lambda}$, or between $\to_{\mathsf{o}}$ and $\to_{\mathsf{o}_\lambda}$) is important for at least two reasons. Firstly, as we have already seen it, if reduction excludes $\to_{\mathsf{e_{var}}}$ then its normal forms have a neat inductive description. Secondly, if reduction excludes $\to_{\mathsf{e_{var}}}$ then it is stable under substitution.

Lemma 3.8 (Stability of reductions without $\to_{\mathsf{e_{var}}}$ under substitution). *Let $a \in \{\mathsf{o}_\lambda, \mathsf{s}_\lambda, \mathsf{vsc}_\lambda\}$. If $t \to_a u$ then $t\{x \leftarrow s\} \to_a u\{x \leftarrow s\}$ for every term s.*

Note that stability under substitution of *values* (that is, when s is a value in the statement of Lemma 3.8) holds also for $\to_{\mathsf{o}}$, $\to_{\mathsf{s}}$ and $\to_{\mathsf{vsc}}$. The problem is that it breaks for $\to_{\mathsf{e_{var}}}$ steps when s is not a value: $(yy)[y \leftarrow x] \to_{\mathsf{oe_{var}}} xx$ but $(yy)[y \leftarrow x]\{x \leftarrow s\} = (yy)[y \leftarrow s] \not\to_{\mathsf{oe_{var}}} ss = (xx)\{x \leftarrow s\}$ if s is not a value. The same remark applies to $\to_{\mathsf{s}}$ and $\to_{\mathsf{s}_\lambda}$, and to $\to_{\mathsf{vsc}}$ and $\to_{\mathsf{vsc}_\lambda}$.

4 PLOTKIN AND SHUFFLING

Here, we compare the VSC with two untyped CbV calculi in the literature, namely Plotkin's [1975] λ_{Plot} and Carraro and Guerrieri's [2014] shuffling calculus. A further case comparison, with Moggi's computational λ-calculus, is in [Accattoli and Guerrieri 2022, Appendix B] for lack of space.

For further relationships, see [Accattoli and Paolini 2012], where the relationship with a calculus by Herbelin and Zimmermann [2009] is studied, or [Accattoli and Guerrieri 2016], where the relationship with the intuitionistic CbV fragment of $\overline{\lambda}\mu\tilde{\mu}$ [Curien and Herbelin 2000] is studied.

Theories. We introduce here the notion of *equational theory*, referred to several calculi, which shall be used for the comparisons of these section and also in the study of collapsibility in Section 6.

Definition 4.1 ((Equational) theories). Let X be a calculus, that is, a set of terms with a binary relation R on it.

- An X-*theory* $\mathcal{T}$ is an equivalence relation containing R and closed by all contexts of X.
- The *equational theory* $\mathcal{T}_{\mathsf{X}}$ of X is the smallest X-theory, that is, it is the symmetric, reflexive, transitive, and contextual closure of R (that is, $\mathcal{T}_{\mathsf{X}}$ is $=_R$ if R is closed by all contexts).

4.1 Plotkin

The original CbV λ-calculus λ_{Plot} [Plotkin 1975] can be easily simulated in the VSC. The syntax of λ_{Plot} is simply the same as in the VSC but without ES. Coherently with our notations, we define $\to_{\mathsf{o}\beta_v}$ and $\to_{\mathsf{f}\beta_v}$ in λ_{Plot} as the closures under open and full contexts (without ES) of the β_v-rule:

$$(\lambda x.t)v \mapsto_{\beta_v} t\{x \leftarrow v\} \qquad \text{where } v \text{ is a value (without ES).}$$

Proposition 4.2 (Simulation). *Let t and t' be terms without ES. If $t \to_{\mathsf{o}\beta_v} t'$ then $t \to_{\mathsf{om}} \cdot \to_{\mathsf{oe}} t'$; and if $t \to_{\mathsf{f}\beta_v} t'$ then $t \to_{\mathsf{m}} \cdot \to_{\mathsf{e}} t'$.*

There is no sensible way to simulate VSC into λ_{Plot}. Indeed VSC is a *proper* extension of λ_{Plot}: terms such as $(\lambda z.\delta)(yy)\delta$ and $\delta((\lambda z.\delta)(yy))$ (with $\delta := \lambda x.xx$) diverge in VSC—and actually $(\lambda x.\delta)(yy)\delta =_{\mathsf{vsc}} \delta((\lambda z.\delta)(yy))$—but in λ_{Plot} they are—*distinct*—$\mathsf{f}\beta_v$-*normal* forms.

Corollary 4.3 (Plotkin $\subsetneq$ VSC). *The equational theory of λ_{Plot} is strictly contained in the equational theory of* VSC*, that is, $\mathcal{T}_{\lambda_{\mathsf{Plot}}} \subsetneq \mathcal{T}_{\mathsf{vsc}}$.*

Despite extending λ_{Plot}, VSC does not lose the CbV essence, as $(\lambda x.y)\Omega$ has no normalizing reduction sequence in both λ_{Plot} and VSC, while in CbN it normalizes in one step, erasing $\Omega := \delta\delta$.

Valuability and Contextual Equivalence. While the equational theory of the VSC is strictly larger than the one of λ_{Plot}, in some respects the two calculi are equivalent, as we now show. First, in the special case where the (open) VSC turns a term into a value v then (the open) λ_{Plot} can do it as well.

Lemma 4.4 (Lifting valuability). *If $t \to_{\mathsf{o}}^{*} v$ and t is without ES, then v is without ES and $t \to_{\mathsf{o}\beta_v}^{*} v$.*

Such a property allows us to show that the contextual equivalences of the two calculi coincide.

Definition 4.5 (CbV contextual equivalence). Let t and u be two terms in a CbV calculus X. We say that t is *contextually equivalent* to u in X, noted $t =_{\mathsf{ce}}^{\mathsf{X}} u$, if for every context C (of X) such that $C\langle t\rangle$ and $C\langle u\rangle$ are closed we have that $C\langle t\rangle \to_{\mathsf{X}}^{*} v$ if and only if $C\langle t\rangle \to_{\mathsf{X}}^{*} v'$, for some values v, v' (of X).

Proposition 4.6 (Equivalence of contextual equivalences). *Let t and u be terms without ES. Then, $t =_{\mathsf{ce}}^{\lambda_{\mathsf{Plot}}} u$ if and only if $t =_{\mathsf{ce}}^{\mathsf{VSC}} u$.*

Proof. Direction $\Rightarrow$ holds because the VSC simulates λ_{Plot} (Prop. 4.2). Direction $\Leftarrow$ follows from the fact that if $t \to_{\mathsf{vsc}}^{*} v$ for some value v then $t \to_{\beta_v} v'$ for some value v'. Indeed, by valuability (Prop. 3.3.1) $t \to_{\mathsf{o}}^{*} v'$ for some value v'; by lifting (Lemma 4.4), v' is without ES and $t \to_{\mathsf{o}\beta_v}^{*} v'$. □

Contextual equivalence shall play a role in Sect. 6. Prop. 4.6 deals with terms with no ES, but what about contextual equivalence on terms with ES? We need a way of expanding ES into β-redexes, that shall be used also in the following sections, and that preserves contextual equivalence.

Definition 4.7 (ES expansion). Given a term t with ES, the expansion of all the ES of t into β-redexes is obtained by applying $\to_{\mathsf{m}}$ backwards, obtaining a term $t^{\bullet}$ without ES. Formally, $(u[x\leftarrow s])^{\bullet} := (\lambda x.u^{\bullet})s^{\bullet}$, and in the other cases $t^{\bullet}$ is defined as expected.

Lemma 4.8 (Stability of contextual equivalence by ES expansion). *Let $t, u \in \Lambda_{\mathsf{vsc}}$: one has $t =_{\mathsf{ce}}^{\mathsf{VSC}} u$ if and only if $t^{\bullet} =_{\mathsf{ce}}^{\mathsf{VSC}} u^{\bullet}$ (if and only if $t^{\bullet} =_{\mathsf{ce}}^{\lambda_{\mathsf{Plot}}} u^{\bullet}$).*

4.2 Shuffling and Structural Equivalence

To relate the VSC to Carraro and Guerrieri's [2014] shuffling calculus λ_{shuf}, we need a concept.

Structural Equivalence. The VSC comes with a notion of *structural equivalence* $\equiv$, that equates terms differing only in the position of ES. A strong justification comes from the CbV linear logic interpretation of λ-terms with ES, in which structurally equivalent terms translate to the same (recursively typed) proof net, see [Accattoli 2015]. Structural equivalence $\equiv$ is defined as the least equivalence relation on terms closed by all contexts and generated by the following root cases:

$$t[x\leftarrow u]s \equiv_{@\mathsf{l}} (ts)[x\leftarrow u] \quad \text{if } x \notin \mathsf{fv}(s) \qquad t[x\leftarrow u[y\leftarrow s]] \equiv_{[\cdot]} t[x\leftarrow u][y\leftarrow s] \quad \text{if } y \notin \mathsf{fv}(t)$$
$$t\,s[x\leftarrow u] \equiv_{@\mathsf{r}} (ts)[x\leftarrow u] \quad \text{if } x \notin \mathsf{fv}(t) \qquad t[y\leftarrow s][x\leftarrow u] \equiv_{\mathsf{com}} t[x\leftarrow u][y\leftarrow s] \quad \text{if } y \notin \mathsf{fv}(u),\, x \notin \mathsf{fv}(s)$$

Pleasantly, adding $\equiv$ results in a smooth system, as $\equiv$ commutes with the rewriting rules, and can thus be postponed. Additionally, the commutation is *strong*, as it preserves the number and kind of steps (thus the cost model)—one says that it is a *strong bisimulation* (with respect to $\to_{\mathsf{vsc}}$). Being a strong bisimulation in particular implies that $\equiv$ is *irrelevant*, as it is the case for $\to_{\mathsf{e}_{\mathsf{var}}}$.

Proposition 4.9 (Operational properties of structural equivalence $\equiv$).

(1) $\equiv$ is a strong bisimulation: *if $t \equiv u$ and $t \to_{\mathsf{a}} t'$ then there exists $u' \in \Lambda_{\mathsf{vsc}}$ such that $u \to_{\mathsf{a}} u'$ and $t' \equiv u'$, for* $\mathsf{a} \in \{\mathsf{m}, \mathsf{e}, \mathsf{e}_{\lambda}, \mathsf{e}_{\mathsf{var}}, \mathsf{vsc}, \mathsf{vsc}_{\lambda}, \mathsf{om}, \mathsf{oe}, \mathsf{oe}_{\lambda}, \mathsf{o}, \mathsf{o}_{\lambda}, \mathsf{sm}, \mathsf{se}, \mathsf{se}_{\lambda}, \mathsf{s}, \mathsf{s}_{\lambda}\}$.

(2) $\equiv$ *is $\to_{\mathsf{a}}$-irrelevant, for* $\mathsf{a} \in \{\mathsf{m}, \mathsf{e}, \mathsf{e}_{\lambda}, \mathsf{e}_{\mathsf{var}}, \mathsf{vsc}, \mathsf{vsc}_{\lambda}, \mathsf{om}, \mathsf{oe}, \mathsf{oe}_{\lambda}, \mathsf{o}, \mathsf{o}_{\lambda}, \mathsf{sm}, \mathsf{se}, \mathsf{se}_{\lambda}, \mathsf{s}, \mathsf{s}_{\lambda}\}$.

Let VSC/$\equiv$ be the VSC endowed with reduction $\to_{\mathsf{vsc}}$ modulo $\equiv$, noted $\to_{\mathsf{vsc}/\equiv}$ and defined as: $t \to_{\mathsf{vsc}/\equiv} u$ if $t \equiv t' \to_{\mathsf{vsc}} u' \equiv u$ for some $t', u' \in \Lambda_{\mathsf{vsc}}$. From Prop. 4.9, it immediately follows that $\to_{\mathsf{vsc}/\equiv}$ is confluent, and that adding $\equiv$ to the VSC does not extend the VSC contextual equivalence.

Corollary 4.10 (Further properties of ≡). *(1) Reduction* $\to_{\mathsf{vsc}/\equiv}$ *is confluent.*
(2) If $t \equiv u$ *then* $t =_{\mathsf{ce}}^{\mathsf{VSC}} u$*. Moreover,* $t =_{\mathsf{ce}}^{\mathsf{VSC}/\equiv} u$ *if and only if* $t =_{\mathsf{ce}}^{\mathsf{VSC}} u$*.*

The same reasoning also applies to any other contextual closure of VSC, with or without $\mapsto_{\mathsf{e_{var}}}$. We shall also show that typability with multi types is invariant by structural equivalence (Prop. 7.4).

Shuffling. The equational theory of Carraro and Guerrieri's [2014; 2015] shuffling calculus λ_{shuf} is contained in VSC/≡. The calculus λ_{shuf} extends Plotkin's [1975] λ_{Plot} with two rules, σ_1 and σ_3:

$$((\lambda x.t)u)s \mapsto_{\sigma_1} (\lambda x.ts)u \qquad\qquad v((\lambda x.s)u) \mapsto_{\sigma_3} (\lambda x.vs)u$$

Proposition 4.11 (Shuffling ⊊ VSC/≡). *The equational theory of* λ_{shuf} *is strictly contained in the one of the* VSC *extended with* ≡*, that is,* $\mathcal{T}_{\lambda_{\mathsf{shuf}}} \subsetneq \mathcal{T}_{\mathsf{vsc}/\equiv}$.

Proof. The containment is proved by Cor. 4.3 for β_v and as follows for σ_1 and σ_3:

$$\begin{array}{l} q := ((\lambda x.t)u)s \mapsto_{\sigma_1} (\lambda x.ts)u =: q' \text{ is captured by } q \to_{\mathsf{m}} t[x\leftarrow u]s \equiv_{@\mathsf{l}} (ts)[x\leftarrow u] \ {}_{\mathsf{m}}\!\leftarrow\ q' \\ q := v((\lambda x.s)u) \mapsto_{\sigma_3} (\lambda x.vs)u =: q' \text{ is captured by } q \to_{\mathsf{m}} vs[x\leftarrow u] \equiv_{@\mathsf{r}} (vs)[x\leftarrow u] \ {}_{\mathsf{m}}\!\leftarrow\ q' \end{array}$$

The following different λ_{shuf}-normal terms are equated in VSC/≡, so the containment is strict:

$$(\lambda y.((\lambda x.z)(zz)))(ww) \to_{\mathsf{m}}^{2} z[x\leftarrow zz][y\leftarrow ww] \equiv_{\mathsf{com}} z[y\leftarrow ww][x\leftarrow zz] \ {}_{\mathsf{m}}^{2}\!\leftarrow (\lambda x.((\lambda y.z)(ww)))(zz) \qquad \square$$

Shape of Inert Terms. Extending VSC with ≡ allows a further simplification of the structure of inert terms (which is however not used in the paper, to confirm the irrelevance of ≡). Because of $\equiv_{@\mathsf{l}}$, substitutions can be grouped together, obtaining that inert terms have the following shape:

$$(xf_1 \dots f_n)[x_1\leftarrow i_1] \dots [x_m\leftarrow i_m] \qquad \text{with } n, m \geq 0.$$

Additionally, by repeatedly applying $\equiv_{@\mathsf{r}}$ and $\equiv_{[\cdot]}$ one can assume that $f_1, \dots, f_n$ and $i_1, \dots, i_m$ do not contain ES at the open level, that is, all their ES (if any) are under abstractions.

5 CALL-BY-VALUE SOLVABILITY AND SCRUTABILITY

In the λ-calculus, the notion of solvability identifies *meaningful* terms. This notion is well studied in the CbN λ-calculus, with an elegant theory, see [Barendregt 1984]. In CbV, as first observed by Paolini and Ronchi Della Rocca [1999; 2004], there are *two* notions that are semantically relevant, solvability and scrutability (which they call *potential valuability*), and neither can be characterized operationally in Plotkin's calculus. Accattoli and Paolini [2012] show that instead the VSC admits natural operational characterizations of both CbV scrutability and solvability.

The definitions of solvability and scrutability depend on the calculus and are *interactive* in the sense they are based on the behavior of a term inside a testing context. For solvability, head contexts are used. The intuition is that they are contexts that cannot discard the plugged term without interacting with it. For scrutability, we need head contexts that additionally cannot turn the plugged term into a value without interacting with it. Such contexts are here called *testing (head) contexts.*

Definition 5.1 (Head context, scrutability, solvability). Let X be a calculus containing the λ-calculus.
A *head context* in X is a context defined by the grammar $H ::= \langle\cdot\rangle \mid \lambda x.H \mid Ht$.
A *testing (head) context* in X is a (head) context defined by the grammar $T ::= \langle\cdot\rangle \mid (\lambda x.T)t \mid Tt$.
A term t in X is X-*scrutable* (or X-*potentially valuable*) if there is a testing context T and a value v in X such that $T\langle t\rangle$ X-reduces to v, and it is X-*inscrutable* otherwise.
A term t in X is X-*solvable* if $H\langle t\rangle$ X-reduces to the identity $\mathsf{I} := \lambda x.x$ for some head context H, and it is X-*unsolvable* otherwise.

Accattoli and Paolini [2012] give characterizations of VSC-solvability and VSC-scrutability akin to Wadsworth's characterization of CbN solvability [Wadsworth 1971, 1976].

Proposition 5.2 (Operational characterization of VSC scrutability/solvability, [Accattoli and Paolini 2012]).

(1) VSC-Scrutability via $\to_{\mathsf{o}}$: *a term t is* VSC*-scrutable if and only if $\to_{\mathsf{o}}$ terminates on t.*
(2) VSC-Solvability via $\to_{\mathsf{s}}$: *a term t is* VSC*-solvable if and only if $\to_{\mathsf{s}}$ terminates on t.*

By irrelevance of $\to_{\mathsf{oe}_{\mathrm{var}}}$ and $\to_{\mathsf{se}_{\mathrm{var}}}$ (Prop. 3.7), the operational characterizations above of VSC-scrutability and VSC-solvability can be reformulated in terms of $\to_{\mathsf{o}_\lambda}$ and $\to_{\mathsf{s}_\lambda}$, respectively.

Corollary 5.3 (Operational characterization of VSC scrutability/solvability, Bis).

(1) VSC-Scrutability via $\to_{\mathsf{o}_\lambda}$: *a term t is* VSC*-scrutable if and only if $\to_{\mathsf{o}_\lambda}$ terminates on t.*
(2) VSC-Solvability via $\to_{\mathsf{s}_\lambda}$: *a term t is* VSC*-solvable if and only if $\to_{\mathsf{s}_\lambda}$ terminates on t.*

Since $\to_{\mathsf{o}} \subsetneq \to_{\mathsf{s}} \subsetneq \to_{\mathsf{vsc}}$ (and $\to_{\mathsf{s}}$ and $\to_{\mathsf{o}}$ are diamond), a consequence of the characterizations in Prop. 5.2 is a sort of *hierarchy* among normalizing, solvable and scrutable terms in the VSC.

Corollary 5.4 (Predicates entailment). *Let t be a term. If t is* vsc*-normalizing, then it is* VSC*-solvable. If t is* VSC*-solvable then it is* VSC*-scrutable.*

Note that the converse implications of Cor. 5.4 fail (see the terms u and s just below). That is, the set of VSC-inscrutable terms is strictly included in the set of VSC-unsolvable terms, which is in turn strictly included in the set of vsc-normalizing terms.

Open reduction captures the fact that $t := \Omega$ (with $\Omega := \delta\delta$ and $\delta := \lambda z.zz$) is VSC-inscrutable, as $\to_{\mathsf{o}}$ diverges on t, while $u := \lambda x.\Omega$ is VSC-scrutable, indeed $\to_{\mathsf{o}}$ terminates on u (as it does not reduce under abstractions). Solving reduction captures the fact that $u := \lambda x.\Omega$ is VSC-unsolvable, as $\to_{\mathsf{s}}$ diverges on u, while $s := x(\lambda x.\Omega)$ is VSC-solvable, indeed $\to_{\mathsf{s}}$ terminates on s, although s is not vsc-normalizing. Note that $(\lambda x.\delta)(yy)\delta$ and $\delta((\lambda x.\delta)(yy))$ are VSC-unsolvable and VSC-inscrutable, while they are normal—but still unsolvable and inscrutable—in Plotkin's calculus.

Scrutability. Paolini and Ronchi Della Rocca [1999; 2004] define scrutability in a slightly different way[4], which is proved to be equivalent to ours in [Accattoli and Guerrieri 2022, Appendix F]. To our knowledge, scrutability has been only studied in CbV, but it also make sense in CbN, where can easily be characterized operationally via *weak* (*i.e.* not reducing under abstractions) head reduction.

Equivalence with Scrutability and Solvability in Plotkin's Calculus. Solvability and scrutability depend on the calculus in which they are defined. Then, what is the relationship between these notions in Plotkin's λ_{Plot} and in the VSC? We show that the two variants of each property coincide, adapting an argument from [Guerrieri et al. 2015, 2017], and that the presence of ES has no impact.

Theorem 5.5 (Robustness of CbV solvability and scrutability). *Let $t \in \Lambda_{\mathsf{vsc}}$ be a term.*

(1) CbV Scrutability: *if t is without ES, then t is* VSC*-scrutable if and only if t is λ_{Plot}-scrutable.*
(2) CbV Solvability: *if t is without ES, then t is* VSC*-solvable if and only if t is λ_{Plot}-solvable.*
(3) ES expansion: *t is* VSC*-scrutable (resp. solvable) if and only if $t^\bullet$ is* VSC*-scrutable (resp. solvable).*

Proof. The right-to-left direction of both Points 1 and 2 is obvious, since $\to^*_{\mathsf{f}\beta_v} \subseteq \to^*_{\mathsf{vsc}}$ (Prop. 4.2). Let us prove the left-to-right directions of Points 1 and 2 separately.

(1) By definition of VSC-scrutability, there is a testing head context T and a value v such that $T\langle t\rangle \to^*_{\mathsf{vsc}} v$. By valuability (Prop. 3.3.1), $T\langle t\rangle \to^*_{\mathsf{o}} v'$ for some value v'. By Lemma 4.4, v' is without ES and $T\langle t\rangle \to^*_{\mathsf{o}\beta_v} v'$. Thus, t is λ_{Plot}-scrutable, since $\to_{\mathsf{o}\beta_v} \subseteq \to_{\mathsf{f}\beta_v}$.

[4] Paolini and Ronchi Della Rocca [1999; 2004] and—with minor variations—[Accattoli and Paolini 2012; Carraro and Guerrieri 2014] define potential valuability (aka scrutability) as follows: t is X-potentially valuable if there are variables $x_1, \dots, x_n$ and values $v, v_1, \dots, v_n$ (with $n \geq 0$) such that the simultaneous substitution $t\{x_1 \leftarrow v_1, \dots, x_n \leftarrow v_n\}$ X-reduces to v.

(2) By definition of VSC-solvability, $s := H\langle t\rangle \to^*_{\mathsf{vsc}} \mathsf{I}$ for some head context H. By valuability (Prop. 3.3.1), $s \to^*_{\mathsf{o}} v$ for some value v. By confluence, $v \to^*_{\mathsf{vsc}} \mathsf{I}$. Clearly, v must be an abstraction $\lambda x.q$ such that $q \to^*_{\mathsf{vsc}} x$, so that $v = \lambda x.q \to^*_{\mathsf{vsc}} \lambda x.x = \mathsf{I}$. Again by valuability, $q \to^*_{\mathsf{o}} v'$ for some value v', and $v' \to^*_{\mathsf{vsc}} x$ by confluence. Note that v' cannot be an abstraction because it would not reduce to a variable. Then $v' = x$. Summing up, $s \to^*_{\mathsf{o}} \lambda x.q$ and $q \to^*_{\mathsf{o}} x$. By lifting valuability (Lemma 4.4), we obtain $s \to^*_{\mathsf{o}\beta_v} \lambda x.q$ and $q \to^*_{\mathsf{o}\beta_v} x$, and putting the two sequences together we obtain $s \to^*_{\mathsf{o}\beta_v} \lambda x.q \to^*_{\mathsf{f}\beta_v} \lambda x.x$, that is, $s \to^*_{\mathsf{f}\beta_v} \mathsf{I}$. Thus, t is λ_{Plot}-solvable.

For the proof of Point 3, see [Accattoli and Guerrieri 2022, Appendix F]. □

For both solvability and scrutability, the equivalence holds also with the VSC extended structural equivalence ≡. This is an easy consequence of the irrelevance of ≡. These results corroborate the idea that solvability and scrutability in CbV are *robust* notions that are independent from the particular CbV calculus used to define them. Thus, we can talk about *CbV solvability* and *CbV scrutability*, instead of X-solvability and X-scrutability for each CbV calculus X. Pushing things even further, one could take Thm. 5.5 as a *criterion* to identify a calculus X extending Plotkin's λ_{Plot} as a *CbV calculus*: the notions of X-solvability and X-scrutability must coincide with those in λ_{Plot}.

Differences between CbV and CbN. Clearly, CbV solvability implies CbN solvability (apply Def. 5.1 to λ_{Plot} and the CbN λ-calculus, noticing that $\to_{\mathsf{f}\beta_v} \subseteq \to_\beta$). However, there is a crucial difference between CbV and CbN solvability: a term such as $x\Omega$ is CbV unsolvable (and $\to_{\mathsf{s}}$ indeed diverges) while it is CbN solvable (it is head normal): in fact, plugging it in a head context can eventually erase Ω in CbN but instead cannot in CbV (similarly, $x\Omega$ is also CbN scrutable but CbV inscrutable).

Proposition 5.6 (CbV solvable strictly implies CbN solvable). *Let t be a term without ES. If t is CbV solvable then it is CbN solvable. Moreover, the converse implication does not hold.*

Orders of (In)Scrutability and Solvability. The relationship between CbV scrutability and CbV solvability is better understood via the following refinement of (in)scrutability by levels, inspired by Longo's [1983] notion of order for λ-terms and yet slightly different.

Definition 5.7 ((In)Scrutability by levels). Let $t \in \Lambda_{\mathsf{vsc}}$ and $n \in \mathbb{N}$. Then *n-(in)scrutability* is defined inductively as follows:

- If t is CbV inscrutable then it is 0-inscrutable.
- If t o_λ-reduces to an inert term then it is 0-scrutable.
- If t o_λ-reduces to a fireball of shape $L\langle\lambda x.u\rangle$ and u is n-inscrutable (resp. n-scrutable) then t is $n+1$-inscrutable (resp. $n+1$-scrutable).

If t is n-scrutable for some $n \in \mathbb{N}$ then it is *finitely (CbV) scrutable*. Additionally, t is *infinitely (CbV) scrutable* if, co-inductively, $\to_{\mathsf{o}_\lambda}$-reduces to a fireball of shape $L\langle\lambda x.u\rangle$ and u is infinitely scrutable.

For instance, Ω is 0-inscrutable, $\lambda x.\Omega$ is 1-inscrutable, x and xy are 0-scrutable, the identity I is 1-scrutable, $\lambda x.\mathsf{I}$ is 2-scrutable. The slight difference with Longo's order can be seen on $\mathsf{I}[x\leftarrow\Omega]$, which is 0-inscrutable despite having a head abstraction. The interesting notion is infinite scrutability, the standard example of which is ZK, where $\mathsf{K} := \lambda x.\lambda y.x$ and $\mathsf{Z} := \lambda y.((\lambda x.y(\lambda z.xxz))(\lambda x.y(\lambda z.xxz)))$ is Plotkin's CbV variant of Curry's fixed point combinator, as it is easily seen that for every $n \in \mathbb{N}$ there is a reduction $\mathsf{ZK} \to^*_{\mathsf{vsc}} t_n$ where t has $2n$ head abstractions. Note that the expected notion of infinite *in*scrutability coincides with infinite scrutability. From the operational characterizations of CbV scrutability and solvability (Cor. 5.3) we obtain the following descriptions, refining Cor. 5.4.

Proposition 5.8 (Dissecting CbV scrutability and solvability via levels). *Let $t \in \Lambda_{\mathsf{vsc}}$:*

(1) t is CbV scrutable if and only if t is either m-inscrutable for some $m > 0$, or finitely scrutable, or infinitely scrutable;

(2) t is CbV solvable if and only if t is finitely scrutable.

Note that in the *scrutable-as-meaningful* approach (which, as we shall see, is the valid one in CbV) infinitely scrutable terms are meaningful, while in the *solvable-as-meaningful* approach (which is the one traditionally adopted in the CbN case) they are unsolvable and thus meaningless.

5.1 Equivalent Definitions

As nicely surveyed by García-Pérez and Nogueira [2016], in CbN there are many equivalent definitions of solvability. Here we focus on three of them, given for a generic calculus X. A term t in X is *solvable* in the sense of SOL-FE, SOL-ID, SOL-EX if respectively:

(1) SOL-FE: *for every* vsc-normal form u there exists a head context H_u such that $H_u\langle t\rangle \to_{\mathrm{X}}^* u$.
(2) SOL-ID: there exists a head context H such that $H\langle t\rangle \to_{\mathrm{X}}^* \mathsf{I}$, where $\mathsf{I} := \lambda x.x$ (the identity).
(3) SOL-EX: *there exists* a vsc-normal form u and a head context H such that $H\langle t\rangle \to_{\mathrm{X}}^* u$.

The implications SOL-FE $\Rightarrow$ SOL-ID $\Rightarrow$ SOL-EX are obvious in every calculus X.

In CbN, the direction SOL-EX $\Rightarrow$ SOL-ID follows easily from the properties of the reduction characterizing solvability (namely, the head normalization theorem, and the stability of head termination by extraction from a head context). Since these properties hold true also for solving reduction (see Prop. 3.5.1 and Prop. 3.5.3 above), the same implication holds in the VSC.

The implication SOL-ID $\Rightarrow$ SOL-FE in CbN is immediate: one has $\mathsf{I}u \to_\beta u$ for every term u, and so if H is the context for SOL-ID then $H_u := Hu$ is the context proving SOL-FE. García-Pérez and Nogueira [2016] stress that, in CbV, $\mathsf{I}u$ does not necessarily reduce to u, if u is not a value. But they do not notice that nonetheless the implication SOL-ID $\Rightarrow$ SOL-FE *does* hold in λ_{Plot} (and thus in the VSC) via a simple argument, pointed out to us by Xavier Montillet and given in the next proof.

Therefore, in the VSC the three definitions of solvability are equivalent, *exactly as in CbN*.

Theorem 5.9 (Equivalent notions of solvability). *In the* VSC*, SOL-EX $\Leftrightarrow$ SOL-ID $\Leftrightarrow$ SOL-FE.*

Proof. The non-trivial implications to prove are SOL-EX $\Rightarrow$ SOL-ID and SOL-ID $\Rightarrow$ SOL-FE.

For SOL-EX $\Rightarrow$ SOL-ID, suppose that t fulfills SOL-EX in VSC, that is, there is a vsc-normal form u and a head context H such that $H\langle t\rangle \to_{\mathrm{vsc}}^* u$. By normalization (Prop. 3.5.1), $H\langle t\rangle \to_{\mathrm{s}}^* s$ for some s-normal s. By stability by extraction from a head context (Prop. 3.5.3), $t \to_{\mathrm{s}}^* q$ for some s-normal q. Then, according to the operational characterization of SOL-ID (Prop. 5.2.2), t verifies SOL-ID.

For SOL-ID $\Rightarrow$ SOL-FE, suppose that t is solvable in the sense of SOL-ID, that is, there is a head context H such that $H\langle t\rangle \to_{\mathrm{vsc}}^* \mathsf{I}$. Let u be a vsc-normal form with $x \notin \mathrm{fv}(u)$ and let $H_u := (H\,\lambda x.u)\mathsf{I}$. Then, $H_u\langle t\rangle \to_{\mathrm{vsc}}^* (\mathsf{I}\lambda x.u)\mathsf{I} \to_{\mathrm{vsc}}^+ (\lambda x.u)\mathsf{I} \to_{\mathrm{vsc}}^+ u$. As H_u is a head context, t verifies SOL-FE. □

One More New Definition. The relevance of inert terms can be stressed by showing that they can be used to provide yet another characterization of CbV solvability.

Theorem 5.10 (Another definition of CbV solvability). *A term t is* VSC*-solvable if and only if*

- SOL-IN*: there is a head context H and an inert term i such that $H\langle t\rangle \to_{\mathrm{vsc}}^* i$.*

Proof. Direction SOL-ID $\Rightarrow$ SOL-IN is straightforward: if H is the context such that $H\langle t\rangle \to_{\mathrm{vsc}}^* \mathsf{I}$ then $H' := Hx$ is such that $H'\langle t\rangle \to_{\mathrm{vsc}}^* \mathsf{I}x \to_{\mathrm{vsc}}^* x$, which is inert. For SOL-IN $\Rightarrow$ SOL-ID, let H be the head context such that $H\langle t\rangle \to_{\mathrm{vsc}}^* i$. Since inert terms are s_λ-normal (Prop. 3.4.3), by the derived operational characterization of SOL-ID (Cor. 5.3.2) there is a context H' such that $H'\langle i\rangle \to_{\mathrm{vsc}}^* \mathsf{I}$. Then the head context $H'\langle H\rangle$ is such that $H'\langle H\langle t\rangle\rangle \to_{\mathrm{vsc}}^* H'\langle i\rangle \to_{\mathrm{vsc}}^* \mathsf{I}$. □

Among the definitions of CbV solvability we discussed, SOL-IN is the only one using as target *open* normal forms (inert terms), and *not* fully normal terms. Thus, solvability can be captured at the open

level, without requiring reduction under abstraction. Indeed, SOL-IN can be equivalently defined using $H\langle t\rangle \to_{o_\lambda}^* i$ instead of $H\langle t\rangle \to_{vsc}^* i$ (proof in [Accattoli and Guerrieri 2022, Appendix F]).

Last, SOL-IN can be adapted to CbN, by replacing inert terms with terms of the form $xt_1 \dots t_n$ with $n \geq 0$ (and no hypotheses on $t_1, \dots, t_n$), sometimes called *neutral terms* (the literature is inconsistent with the terminology, at times the definition of neutral terms requires $t_1, \dots, t_n$ to be normal). This fact is both positive and negative: it is good that the open characterization can be adapted, but it shows that the open characterization depends on the calculus (inert/neutral terms in CbV/CbN), while SOL-ID is calculus-independent.

6 (NON-)COLLAPSIBILITY

In CbN, unsolvable terms are *collapsible*, that is, the equational theory $\mathcal{H}$, extending β-conversion by equating all unsolvable terms, is *consistent*, *i.e.* it does not equate all terms. Here we show that CbV inscrutable terms are collapsible, while CbV unsolvable terms are not. This section mostly adapts results from Egidi et al. [1992], presenting them in a different way.

CbV Inscrutable Terms Are Collapsible. The collapsibility of inscrutable terms is obtained by exhibiting a consistent theory that equates them, namely CbV contextual equivalence (Def. 4.5).

Showing that $=_{ce}^{\lambda_{Plot}}$ is a λ_{Plot}-theory, and that $=_{ce}^{VSC}$ is a VSC-theory—see Def. 4.1—is effortless, in particular context closure follows immediately from the definition of contextual equivalence.

Proposition 6.1 (Consistency of CbV contextual equivalence). *CbV contextual equivalence is consistent in both Plotkin's calculus λ_{Plot} and the VSC.*

Proof. Simply note that $\Omega \neq_{ce} \lambda x.\Omega$ in λ_{Plot} and VSC, since the two terms are closed and the empty context distinguishes them: $\lambda x.\Omega$ reduces to a value (itself) in 0 steps, while Ω diverges. □

In analogy to the CbN *sensible theories*, which are λ-theories collapsing all CbN unsolvable terms, and *semi-sensible* ones, which do not equate solvable and unsolvable terms (see [Barendregt 1984]), we introduce the corresponding scrutable notions for CbV.

Definition 6.2 (Scrutable theories). A λ_{Plot}-theory (resp. VSC-theory) is *scrutable* if it equates all CbV inscrutable terms without ES (resp. terms in Λ_{vsc}) and *semi-scrutable* if it does not equate CbV scrutable and inscrutable terms without ES (resp. terms in Λ_{vsc}).

The fact that contextual equivalence in λ_{Plot} is a scrutable theory easily follows from a result in the literature, the full abstraction of CbV applicative bisimilarity (Egidi et al. [1992]; Pitts [2012]).

Proposition 6.3 (λ_{Plot} contextual equivalence is scrutable). *$=_{ce}^{\lambda_{Plot}}$ is a scrutable λ_{Plot}-theory.*

From Prop. 6.3 and the fact that the expansion of ES preserves contextual equivalence (Lemma 4.8) and CbV scrutability (Thm. 5.5.3), it follows that contextual equivalence in VSC is a scrutable theory.

Corollary 6.4 (VSC contextual equivalence is scrutable). *$=_{ce}^{VSC}$ is a scrutable VSC-theory.*

CbV Unsolvable Terms Are Not Collapsible. Perhaps surprisingly, in CbV unsolvable terms are not collapsible. This crucial fact is referred to by García-Pérez and Nogueira [2016] by pointing to Paolini and Ronchi Della Rocca [1999], where however it is not stated. To our knowledge, it is never formally stated anywhere in the literature, which is why we present it here. The argument in the next theorem adapts the idea in the proof by Egidi et al. [1992] that λ_{Plot} contextual equivalence $=_{ce}^{\lambda_{Plot}}$ is a maximal consistent λ_{Plot}-theory (Proposition 35, therein).

Theorem 6.5 (Non-collapsibility of unsolvable terms).

(1) Any scrutable λ_{Plot}-theory (or VSC-theory) $\mathcal{T}$ that is not semi-scrutable is inconsistent.

(2) The set of CbV unsolvable terms is not collapsible.

Proof. (1) Since $\mathcal{T}$ is not semi-scrutable, there are t CbV scrutable and u CbV inscrutable such that $t =_{\mathcal{T}} u$. Since t is CbV scrutable, there is a testing context T sending it to a value v. Since u is CbV inscrutable, $T\langle u\rangle$ is also CbV inscrutable (as the composition $T'\langle T\rangle$ of two testing contexts T, T' is a testing context). By the definition of λ_{Plot}-theory, we have $T\langle u\rangle =_{\mathcal{T}} T\langle t\rangle =_{\mathcal{T}} v$. Now, let s be a term and $y \notin \mathrm{fv}(s)$. Then $s =_{\mathcal{T}} (\lambda y.s)v$ because $=_{\beta_v} \subseteq \mathcal{T}$ by definition of λ_{Plot}-theory. By the context closure of theories and $T\langle u\rangle =_{\mathcal{T}} v$, we obtain $(\lambda y.s)v =_{\mathcal{T}} (\lambda y.s)T\langle u\rangle$. Since $\mathcal{T}$ is scrutable and both $T\langle u\rangle$ and $(\lambda y.s)T\langle u\rangle$ are CbV inscrutable, $(\lambda y.s)T\langle u\rangle =_{\mathcal{T}} T\langle u\rangle$. Therefore, $s =_{\mathcal{T}} T\langle u\rangle$ for every term s, that is, $\mathcal{T}$ is inconsistent.

(2) Any λ_{Plot}-theory $\mathcal{T}$ equating all CbV unsolvable terms is scrutable (since any CbV inscrutable term is CbV unsolvable, Cor. 5.4) but not semi-scrutable, as $\Omega =_{\mathcal{T}} \lambda x.\Omega$ (both terms are CbV unsolvable) with Ω CbV inscrutable and $\lambda x.\Omega$ CbV scrutable. By Point 1, $\mathcal{T}$ is inconsistent. □

(Non-)Genericity. In CbN, unsolvable terms have the *genericity* property [Barendregt 1984, Prop. 14.3.24], which intuitively states that if a term normalizes despite having an unsolvable subterm, then that subterm is not used during the normalization process and can be replaced by any term. In CbV, genericity is the following statement: *if $t \in \Lambda_{\mathsf{vsc}}$ is CbV unsolvable and $F\langle t\rangle \to^*_{\mathsf{vsc}} u$ with u* vsc*-normal and F full context, then $F\langle s\rangle \to^*_{\mathsf{vsc}} u$ for all $s \in \Lambda_{\mathsf{vsc}}$*. This property does *not* hold in CbV, there is a counterexample. Take $t := \lambda y.\Omega$, $F := (\lambda z.x)\langle\cdot\rangle$, and $u := x$. They satisfy the hypotheses of the statement: t is CbV unsolvable, u is a vsc-normal form, and $F\langle t\rangle = (\lambda z.x)(\lambda y.\Omega) \to^*_{\mathsf{vsc}} x = u$. Now, take $s := \Omega$ and note that $F\langle s\rangle = (\lambda z.x)\Omega$ does not reduce to x, because Ω is not (and does not reduce to) a value and so cannot be erased. Hence, genericity for CbV *unsolvable* terms is false.

We claim that genericity does hold for CbV *inscrutable* terms (we do not prove it here, as easy techniques for genericity such as the one by Takahashi [1994] do not apply to the case of inscrutable terms), and even that a generalized genericity property does hold for n-inscrutable terms, along the lines of the *partial genericity lemma* of García-Pérez and Nogueira [2016].

6.1 Axioms for Collapsibility

Kennaway et al. [1999] provide three axioms in order for a set of terms U of the λ-calculus to be collapsible and also satisfy a genericity property[5]. Because of the unusual rewrite rules at a distance of the VSC, it is unclear if it fits into the class of rewriting systems covered by Kennaway et al.'s [1999] axiomatics, which is not clearly specified. It is nonetheless instructive to see how the axioms are instantiated in our setting by taking U as the set of either inscrutable or unsolvable terms.

Axiom 1. The first axiom asks the stability of the terms in U by substitution, that is, *if $t \in U$ then $t\{x\leftarrow u\} \in U$ for every u*. In our setting, both inscrutable and unsolvable terms verify this axiom.

Proposition 6.6 (Stability of CbV scrutability/solvability under removal). *If there exist u such that $t\{x\leftarrow u\}$ is CbV scrutable (resp. solvable) then t is CbV scrutable (resp. solvable).*

Proof. By contradiction, suppose that t is CbV inscrutable. According to the operational characterization of CbV scrutability (Cor. 5.3.1), $\to_{\mathsf{o}_\lambda}$ diverges on t. By stability of $\to_{\mathsf{o}_\lambda}$ by substitution (Lemma 3.8), $\to_{\mathsf{o}_\lambda}$ diverges on $t\{x\leftarrow u\}$ for every term u, so $t\{x\leftarrow u\}$ is CbV inscrutable by Cor. 5.3.1.

The proof concerning CbV solvability is analogous, just replace the properties for $\to_{\mathsf{o}_\lambda}$ with their analogue for $\to_{\mathsf{s}_\lambda}$, in particular Cor. 5.3.1 with Cor. 5.3.2 □

The proof of Prop. 6.6 relies on the stability under substitution for $\to_{\mathsf{o}_\lambda}$ and $\to_{\mathsf{s}_\lambda}$. Note that, as we have seen in Section 3.5, such a property *fails* instead for $\to_{\mathsf{e}_{\mathsf{var}}}$ steps. Therefore, Prop. 6.6 is a

[5]Their notion of genericity however is not equivalent to the one in [Barendregt 1984]: in [Kennaway et al. 1999] plugging in a context—which is part of the statement of genericity—is a capture-avoiding operation, while for Barendregt it is not.

point where the irrelevance of $\to_{e_{var}}$ plays a crucial role. Note also that, as pointed out in Sect. 1, (CbN) diverging terms are not collapsible because they are not stable by substitution, that is, they violate axiom 1. While CbV unsolvable terms are also not collapsible, they do satisfy axiom 1.

Axiom 2. The second axiom is the stability of terms in U by reduction, which in our setting is an easy consequence of the normalization theorem for open/solving reduction (Props. 3.3.2 and 3.5.1).

Lemma 6.7 (Stability of unsolvable terms by reduction). *Let t be CbV inscrutable (resp. unsolvable) and $t \to_{vsc} u$. Then u is CbV inscrutable (resp. unsolvable).*

Proof. By contradiction. If u is CbV scrutable then by the operational characterization of CbV scrutability (Prop. 5.2.1), $u \to_o^* s$ for some o-normal s. Then $t \to_{vsc}^* s$. By the normalization property for $\to_o$ (Prop. 3.3.2), $\to_o$ terminates on t, which then is CbV scrutable (Prop. 5.2.1 again)—absurd.

The proof concerning CbV (in)solvability is analogous, just replace the properties for $\to_o$ with their analogue for $\to_s$: Props. 5.2.1 and 3.3.2 with Props. 5.2.2 and 3.5.1, respectively. □

Axiom 3. The third axiom is more technical and about overlappings of redex patterns with terms in U. Roughly, in our case it amounts to prove that in the two root rules:

$$L\langle\lambda x.t\rangle u \mapsto_m L\langle t[x\leftarrow u]\rangle \qquad\qquad t[x\leftarrow L\langle v\rangle] \mapsto_e L\langle t\{x\leftarrow v\}\rangle$$

if $L\langle\lambda x.t\rangle \in U$ then $L\langle\lambda x.t\rangle u \in U$, and if $L\langle v\rangle \in U$ then $t[x\leftarrow L\langle v\rangle] \in U$. Interestingly, both conditions hold when taking as U the set of CbV inscrutable terms, while the second one *fails* for CbV unsolvable terms. A counterexample is obtained by taking the CbV unsolvable term $\lambda y.\Omega$ and noting that $\mathsf{I}[x\leftarrow\lambda y.\Omega] \mapsto_e \mathsf{I}$ is instead CbV solvable. This fact recasts in Kennaway et al.'s [1999] axiomatics the non-collapsibility of CbV unsolvable terms (Thm. 6.5).

7 MULTI TYPES BY VALUE

This section starts the second part of the paper, where the VSC is studied via a *multi type system*. We first recall the background about multi types and provide an overview of our results.

7.1 From Multi Types to Call-by-Value Solvability

Intersection types are a standard and flexible tool to study λ-calculi, mainly used to characterize termination properties [Coppo and Dezani-Ciancaglini 1978, 1980; Krivine 1993; Pottinger 1980], as well as to study λ-models [Abramsky 1991; Barendregt et al. 1983; Coppo et al. 1987; Egidi et al. 1992; Honsell and Rocca 1992; Plotkin 1993].

The *non-idempotent* variant of intersection types, where the intersection $A \cap A$ is not equivalent to A, was introduced by Gardner [1994]. Then Kfoury [2000], Neergaard and Mairson [2004], and de Carvalho [2007, 2018] provided a first wave of works about them. A survey can be found in Bucciarelli et al. [2017]. Non-idempotent intersections can be seen as *multisets*, which is why, to ease the language, we prefer to call them *multi types* rather than *non-idempotent intersection types*. Multi types refine intersection types with multiplicities, giving rise to a *quantitative* approach that reflects resource consumption, and that it turns out to coincide exactly with the one at work in Girard's [1987] linear logic. Neergaard and Mairson [2004] prove that type inference for multi types is equivalent to normalization. Therefore, multi types hide a computational mechanism.

De Carvalho's Bounds from Multi Types. An insightful use of multi types and of their computational mechanism is de Carvalho's extraction of bounds for the CbN λ-calculus [de Carvalho 2007, 2018]: from certain type derivations, he extracts *exact* bounds about the length of reduction sequences and the size of the normal form of a term, according to various notions of reduction. In particular, for *head* reduction, which in CbN is the reduction characterizing solvability. De Carvalho's seminal work has been extended to many notions of reduction and formalisms. A first wave was inspired

directly from his original work [Bernadet and Lengrand 2013; de Carvalho et al. 2011; de Carvalho and Tortora de Falco 2016; Guerrieri 2019; Manzonetto et al. 2019], and a second wave [Accattoli et al. 2021b,c; Accattoli and Guerrieri 2018; Accattoli et al. 2019b; Alves et al. 2019; Bucciarelli et al. 2020; Dal Lago et al. 2021; Kesner et al. 2021; Kesner and Vial 2020; Kesner and Viso 2022] started after the revisitation of de Carvalho's technique by Accattoli et al. [2018].

Closed CbV and Multi Types. Ehrhard [2012] introduces a CbV system of multi types to study Plotkin's λ_{Plot} with closed terms. It is the CbV version of the system for CbN [de Carvalho 2007, 2018; Gardner 1994]. Both systems can be seen as the restrictions of the relational semantics of linear logic [Bucciarelli and Ehrhard 2001; Girard 1988] to the CbN/CbV translations of the λ-calculus.

Open CbV and Multi Types. Accattoli and Guerrieri [2018]; Guerrieri [2019] use Ehrhard's system to study *Open CbV* (that is, weak call-by-value with possibly open terms). They show that the open reduction of a term t terminates in Open CbV if and only if t is typable with CbV multi types. Moreover, they show how to extract exact bounds from type derivations, adapting de Carvalho's technique. Since termination of open reduction characterizes CbV scrutability (Prop. 5.2.1), their results provide a quantitative characterization of CbV scrutability via multi types.

CbV Solvability and Multi Types, Qualitatively. Here, we build over their work, using Ehrhard's CbV multi types to study Accattoli and Paolini's solving reduction. Since solving reduction *extends* open reduction, the terms that are solving terminating—that is, solvable terms—form a subset of the open terminating ones and so cannot be characterized simply as the typable ones. We characterize them as those typable with certain *solvable types*, inspired by Paolini and Ronchi Della Rocca [1999] and at the same time fixing some technical issues of similar characterizations in [Kerinec et al. 2021; Paolini and Ronchi Della Rocca 1999] (see [Accattoli and Guerrieri 2022, Appendix A] for details).

CbV Solvability and Multi Types, Quantitatively. A further contribution is that, for the first time in the literature, we provide a *quantitative* characterization of CbV solvability, adapting once more de Carvalho's technique. First, we show that every solvable derivation provides bounds to the length of solving reduction sequences *and* the size of the solving normal form. Second, we characterize solvable derivations that provide *exact* bounds. This last part requires introducing *two* refinements of solvable types, detailed in Section 9.

7.2 Introducing Multi Types by Value

Multi Types. There are two mutually defined layers of types, *linear* and *multi types*, their grammars are in Figure 2. We use X for a fixed unspecified ground type, and $[A_1, \dots, A_n]$ is our notation for finite multisets. The *empty* multi type $[\,]$ (obtained taking $n = 0$) is also denoted by $\mathbf{0}$. A multi type is *ground* if it is of the form $n[X] := [X, \dots, X]$ (n times X) for some $n \geq 0$ (so, $0[X] = \mathbf{0}$). A generic (multi or linear) type is noted T. A multi type $[A_1, \dots, A_n]$ has to be intended as a conjunction $A_1 \cap \cdots \cap A_n$ of linear types $A_1, \dots, A_n$, for a commutative, associative, non-idempotent conjunction $\cap$ (morally a tensor $\otimes$), whose neutral element is $\mathbf{0}$.

Intuitively, a linear type corresponds to a single use of a term t, and t is typed with a multiset M of n linear types if it is going to be used (at most) n times. The meaning of *using a term* is not easy to define precisely. Roughly, it means that if t is part of a larger term u, then (at most) n copies of t shall end up in evaluation positions—where they are applied to some terms—while evaluating u.

The derivation rules for the multi types system are in Figure 2 (explanation follows). The rules are the same as in [Ehrhard 2012], up to the extension to ES. A *multi* (resp. *linear*) *judgment* has the shape $\Gamma \vdash t : T$ where t is a term, T is a multi (resp. linear) type, and Γ is a *type context*, that is, a total function from variables to multi types such that the set $\mathrm{dom}(\Gamma) := \{x \mid \Gamma(x) \neq \mathbf{0}\}$ is finite.

$$\text{Linear types } A,B ::= X \mid M \multimap N \qquad \text{Multi types } M,N ::= [A_1,\dots,A_n] \quad n \geq 0$$

$$\frac{}{x:[A] \vdash x:A}\,\mathsf{ax} \qquad \frac{\Gamma, x:M \vdash t:N}{\Gamma \vdash \lambda x.t : M \multimap N}\,\lambda \qquad \frac{[\Gamma_i \vdash v:A_i]_{i\in I} \quad I \text{ finite}}{\uplus_{i\in I}\Gamma_i \vdash v : \uplus_{i\in I}[A_i]}\,\mathsf{many}$$

$$\frac{\Gamma \vdash t:[M\multimap N] \quad \Delta \vdash u:M}{\Gamma \uplus \Delta \vdash tu:N}\,@ \qquad \frac{\Gamma, x:M \vdash t:N \quad \Delta \vdash u:M}{\Gamma \uplus \Delta \vdash t[x\leftarrow u]:N}\,\mathsf{es}$$

Fig. 2. Call-by-Value Multi Type System for VSC.

Explanations about the Inference Rules. All rules but ax and λ assign a multi type to the term on the right-hand side of a judgment. Values are the only terms that can be typed by a linear type, via ax and λ. Rule many can be applied only to values, turning linear types into multi types: it has as many premises as the elements in the (possibly empty) set of indices I (when $I = \emptyset$, the rule has no premises, and it gives an empty multi type $\mathbf{0}$). Note that *every* value can then be typed with $\mathbf{0}$. The many rule says how many "copies" of one occurrence of a value in a term t are needed to evaluate t. It corresponds to the promotion rule of Girard's [1987] linear logic, which, in the CbV representation of the λ-calculus, is indeed used for typing values.

Example of Type Derivation. Let $M := [[X] \multimap [X]]$. Consider the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x:[M\multimap M]\vdash x:M\multimap M}\,\mathsf{ax}}{x:[M\multimap M]\vdash x:[M\multimap M]}\,\mathsf{many} \quad \dfrac{\dfrac{}{x:M\vdash x:[X]\multimap[X]}\,\mathsf{ax}}{x:M\vdash x:M}\,\mathsf{many}}{x:[M\multimap M]\uplus M\vdash xx:M}\,@}{\vdash \lambda x.xx:([M\multimap M]\uplus M)\multimap M}\,\lambda}{\vdash \lambda x.xx:[([M\multimap M]\uplus M)\multimap M]}\,\mathsf{many} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{y:M\vdash y:[X]\multimap[X]}\,\mathsf{ax}}{y:M\vdash y:M}\,\mathsf{many}}{\vdash \mathsf{I}:M\multimap M}\,\lambda \quad \dfrac{\dfrac{\dfrac{}{y:[X]\vdash y:X}\,\mathsf{ax}}{y:[X]\vdash y:[X]}\,\mathsf{many}}{\vdash \mathsf{I}:[X]\multimap[X]}\,\lambda}{\vdash \mathsf{I}:[M\multimap M]\uplus M}\,\mathsf{many}}{\vdash (\lambda x.xx)\mathsf{I}:M}\,@$$

Note that the argument identity $\mathsf{I} := \lambda y.y$ is typed *twice*, and with different types. It is typed once with $[M \multimap M]$, when it is used as a function, and once with M, when it is used as a value. Thus, multi types account for a form of finite polymorphism. Moreover, the finite polymorphism of multi types allows us to type the term $\lambda x.xx$, which is not typable with simple types.

Technicalities about Types. The type context Γ is *empty* if $\mathrm{dom}(\Gamma) = \emptyset$. *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 $J = \emptyset$. A type context Γ is denoted by $x_1:M_1,\dots,x_n:M_n$ (for some $n \in \mathbb{N}$) if $\mathrm{dom}(\Gamma) \subseteq \{x_1,\dots,x_n\}$ and $\Gamma(x_i) = M_i$ for all $1 \leq i \leq n$. Given two type contexts Γ and Δ such that $\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Delta) = \emptyset$, the type context Γ, Δ is defined by $(\Gamma,\Delta)(x) := \Gamma(x)$ if $x \in \mathrm{dom}(\Gamma)$, $(\Gamma,\Delta)(x) := \Delta(x)$ if $x \in \mathrm{dom}(\Delta)$, and $(\Gamma,\Delta)(x) := \mathbf{0}$ otherwise. Note that $\Gamma, x:\mathbf{0} = \Gamma$, where we implicitly assume $x \notin \mathrm{dom}(\Gamma)$.

We write $\Phi \rhd \Gamma \vdash t:M$ if Φ is a (*type*) *derivation* (*i.e.* a tree built up from the rules in Figure 2) with conclusion the multi judgment $\Gamma \vdash t:M$. In particular, we write $\Phi \rhd \vdash t:M$ when Γ is empty. We write $\Phi \rhd t$ if $\Phi \rhd \Gamma \vdash t:M$ for some type context Γ and multi type M.

The Sizes of Type Derivations. Our study being quantitative, we need a notion of size of type derivations. In fact, we shall use *two* notions of size.

Definition 7.1 (Derivation size(s)). Let Φ be a derivation. The *(general) size* $|\Phi|$ of Φ is the number of rule occurrences in Φ except for the rule many. The *multiplicative size* $|\Phi|_{\mathsf{m}}$ of Φ is the number of occurrences of the rules λ and @ in Φ.

The two sizes for derivations play different roles. Qualitatively, to prove that typability implies termination of open/solving reduction, we need a measure that decreases for all open/solving steps;

this role is played by the general size $|\cdot|$. Quantitatively, we want to measure the number of $\to_{\mathsf{m}}$ steps in open/solving reduction sequences, because it is the time cost model of VSC, see [Accattoli et al. 2021a]; this role is played by the multiplicative size $|\cdot|_{\mathsf{m}}$.

Substitution and Removal. Two lemmas establish a key feature of this type system: in a typed term t, substituting a value for a variable as in the exponential step, or, dually, removing a value, preserves the type of t and *consumes* (dually, *adds*) the multi type of the variable. The lemmas also provide quantitative information about the type derivation for t before and after the substitution/removal.

Lemma 7.2 (Substitution). *Let t be a term, v be a value. If $\Phi \rhd \Gamma, x:N \vdash t:M$ and $\Psi \rhd \Delta \vdash v:N$, then there is a derivation $\Theta \rhd \Gamma \uplus \Delta \vdash t\{x\leftarrow v\}:M$ with $|\Theta|_{\mathsf{m}} = |\Phi|_{\mathsf{m}} + |\Psi|_{\mathsf{m}}$ and $|\Theta| \leq |\Phi| + |\Psi|$.*

Lemma 7.3 (Removal). *Let t be a term, v be a value. If $\Phi \rhd \Gamma \vdash t\{x\leftarrow v\}:M$ then there are derivations $\Psi \rhd \Delta, x:N \vdash t:M$ and $\Theta \rhd \Sigma \vdash v:N$ such that $\Gamma = \Delta \uplus \Sigma$, $|\Phi|_{\mathsf{m}} = |\Psi|_{\mathsf{m}} + |\Theta|_{\mathsf{m}}$ and $|\Phi| \leq |\Psi| + |\Theta|$.*

Lemmas 7.2 and 7.3 are needed to prove subject reduction and expansion, respectively, which mean that the type is preserved after and before any reduction step. It holds not only for $\to_{\mathsf{vsc}}$ but also for $\equiv$. Here we state a *qualitative* version. Quantitative versions of subject reduction are in the next sections, they hold for some restrictions of the reduction.

Proposition 7.4 (Qualitative subject reduction and expansion). *Let t $(\to_{\mathsf{vsc}} \cup \equiv)$ t'. There is a derivation $\Phi \rhd \Gamma \vdash t:M$ if and only if there is a derivation $\Phi' \rhd \Gamma \vdash t':M$.*

By Prop. 7.4, our type system does not suffer from Kesner's counterexample (see [Accattoli and Guerrieri 2022, Appendix A.2]) to subject reduction for the type system of Kerinec et al. [2021]. Indeed the counterexample concerns step σ_3, which is subsumed by $\to_{\mathsf{vsc}} \cup \equiv$ (Prop. 4.11), and for which Prop. 7.4 proves subject reduction.

The Special Role of Inert Terms. In the characterizations via multi types of the following two sections, inert terms play a crucial role. In statements about solvable normal forms, they usually satisfy stronger properties, essential for the induction to go through.

8 MULTI TYPES FOR OPEN CBV

Here we recall the relationship between CbV multi types and Open CbV developed by Accattoli and Guerrieri [2018]; Guerrieri [2019]. The reason is threefold:

(1) *Building block*: the solvable case of the next section relies on the open one, because solving reduction is an iteration under head abstractions of open reduction.
(2) *Blueprint*: the open case provides the blueprint for the solvable case.
(3) *Adapting a few details*: the development in [Accattoli and Guerrieri 2018] needs to be slightly adapted to our present framework. Namely, here we use the Open VSC instead of the *split fireball calculus* used there (another formalism for Open CbV), and we include a *ground type* X—absent there—required to deal with solving reduction in the next section.

The Open Size of Terms. For our quantitative study, we need a notion of term size. We actually need a term size for *each* notion of reduction (open here, solving in the next section) that we aim at measuring via multi types. Basically, the size counts the constructors of a term that can be traversed by the reduction. The *open size* $|t|_{\mathsf{o}}$ of a term t, then, is its number of applications out of abstractions:

$$|x|_{\mathsf{o}} := 0 \qquad |\lambda x.t|_{\mathsf{o}} := 0 \qquad |tu|_{\mathsf{o}} := |t|_{\mathsf{o}} + |u|_{\mathsf{o}} + 1 \qquad |t[x\leftarrow u]|_{\mathsf{o}} := |t|_{\mathsf{o}} + |u|_{\mathsf{o}}.$$

Overview of the Characterization. Qualitatively, the open reduction of t terminates if and only if t is typable. Since $\to_{\mathsf{o}}$ does not reduce under abstractions, every abstraction is o-normal (even CbV

unsolvable ones) and hence must be typable: for this reason, $\lambda x.\delta\delta$ ($\delta \coloneqq \lambda x.xx$) is typable with $\mathbf{0}$ (take the derivation only made of one rule many with 0 premises), though $\delta\delta$ is not.

Quantitatively, the multiplicative size $|\Phi|_{\mathsf{m}}$ of *every* type derivation Φ for t provides *upper* bounds to the sum of the length of the open reduction of t plus the open size of its open normal form. To obtain *exact* bounds, one has to avoid typing parts of the term that cannot be touched by open reduction, that is, the body of abstractions (out of other abstractions). Types control in different ways the possibly many abstractions of an inert term or a term that is itself an abstraction. The former is controlled by the typing context, via a *inert* predicate, the latter by the right-hand type, which needs to not be an arrow type. For the constraint to hold for fireballs, independently of whether they are inert terms or values, the two constraint are put together in the *tight* predicate.

Inert and Tight Derivations. Inert types are defined as follows, with $n \geq 0$.

$$\text{Inert multi type } M^{\mathsf{i}} ::= [A^{\mathsf{i}}_1, \dots, A^{\mathsf{i}}_n] \qquad \text{Inert linear type } A^{\mathsf{i}} ::= X \mid n[X] \multimap M^{\mathsf{i}}$$

Note that every ground multi type $n[X]$ is inert.

Definition 8.1 (Inert and tight derivations). A type context $\Gamma = x_1 : M_1, \dots, x_n : M_n$ is *inert* if $M_1, \dots, M_n$ are inert multi types. A derivation $\Phi \rhd \Gamma \vdash t : M$ is *inert* if Γ is an inert type context, and it is *tight* if moreover M is ground.

Note that the definitions of inert and tight derivations depend only on their final judgment.

The next lemma states the first key property of inert terms, that the inertness of their typing context spreads to the right-hand type. It is used to propagate inertness and tightness from the final judgment to the internal ones, allowing us to apply the *i.h.* in proofs.

Lemma 8.2 (Spreading of inertness on judgments). *Let $\Phi \rhd \Gamma \vdash i : M$ be a inert derivation and i be an inert term. Then, M is a inert multi type.*

Correctness. Open correctness establishes that all typable terms o-normalize and the multiplicative size of the derivation bounds the number of $\to_{\mathsf{om}}$ steps plus the open size of the o-normal form; this bound is exact if the derivation is tight. Open correctness is proved following a standard scheme in two stages: (*a*) *quantitative* subject reduction states that every $\to_{\mathsf{o}}$ step preserves types and decreases the general size of a derivation, and that any $\to_{\mathsf{om}}$ step decreases by an exact quantity the multiplicative size of a derivation; (*b*) a lemma states that the multiplicative size of any derivation typing a o-normal form t provides an *upper bound* to the open size of t, and if moreover the derivation is tight then the bound is *exact*.

Lemma 8.3 (Size of fireballs). *Let $\Phi \rhd \Gamma \vdash t : M$ be a derivation.*

(1) If $t = i$ is an inert term then $|\Phi|_{\mathsf{m}} \geq |i|_{\mathsf{o}}$. If moreover Φ is inert, then $|\Phi|_{\mathsf{m}} = |i|_{\mathsf{o}}$.
(2) If $t = f$ is a fireball then $|\Phi|_{\mathsf{m}} \geq |f|_{\mathsf{o}}$. If moreover Φ is tight, then $|\Phi|_{\mathsf{m}} = |f|_{\mathsf{o}}$.

Note that for inert terms the equality of sizes is ensured by the weaker inert predicate. Let us show how tightness enforces the equality of sizes. We have that $\delta \coloneqq \lambda x.xx$ is typable, has size $|\delta|_{\mathsf{o}} = 0$, and any derivation $\Phi \rhd \Gamma \vdash \delta : M$ ends with rule many. If M is not ground (and Φ not tight) then many has at least one premise that types the subterm xx, so $|\Phi|_{\mathsf{m}} > 0 = |\delta|_{\mathsf{o}}$. If M is ground, then $M = \mathbf{0}$ and many has no premises, that is, $|\Phi|_{\mathsf{m}} = 0 = |\delta|_{\mathsf{o}}$.

Now, we can prove quantitative subject reduction, from which open correctness follows. Note that quantitative subject reduction does not need the inert nor the tight predicate.

Proposition 8.4 (Open quantitative subject reduction). *Let $\Phi \rhd \Gamma \vdash t : M$ be a derivation.*

(1) Multiplicative step: *if $t \to_{\mathsf{om}} t'$ then there is $\Phi' \rhd \Gamma \vdash t' : M$ with $|\Phi'|_{\mathsf{m}} = |\Phi|_{\mathsf{m}} - 2$, $|\Phi'| = |\Phi| - 1$;*
(2) Exponential step: *if $t \to_{\mathsf{oe}} t'$ then there is $\Phi' \rhd \Gamma \vdash t' : M$ with $|\Phi'|_{\mathsf{m}} = |\Phi|_{\mathsf{m}}$ and $|\Phi'| < |\Phi|$.*

Solvable multi type	$M^{\mathsf{s}} ::= [A^{\mathsf{s}}_1, \dots, A^{\mathsf{s}}_n] \;\; n > 0$	Solvable linear type	$A^{\mathsf{s}} ::= X \mid M \multimap M^{\mathsf{s}}$
Unitary s. multi type	$M^{\mathsf{us}} ::= [A^{\mathsf{us}}]$	Unitary s. linear type	$A^{\mathsf{us}} ::= X \mid M \multimap M^{\mathsf{us}}$
Inertly s. multi type	$M^{\mathsf{is}} ::= [A^{\mathsf{is}}_1, \dots, A^{\mathsf{is}}_n] \;\; n > 0$	Inertly s. linear type	$A^{\mathsf{is}} ::= X \mid M^{\mathsf{i}} \multimap M^{\mathsf{is}}$

Fig. 3. Kinds of solvable types. A type is *precisely solvable* if it is unitary and inertly solvable.

Theorem 8.5 (Open correctness). *Let $\Phi \rhd \Gamma \vdash t : M$. Then there is a* o*-normalizing reduction $d : t \to^*_{\mathsf{o}} u$ with $2|d|_{\mathsf{m}} + |u|_{\mathsf{o}} \leq |\Phi|_{\mathsf{m}}$. And if Φ is tight, then $2|d|_{\mathsf{m}} + |u|_{\mathsf{o}} = |\Phi|_{\mathsf{m}}$.*

By the operational characterization of CbV scrutability (Prop. 5.2.1), open correctness says in particular that *only* VSC-scrutable terms are typable (with a multi type).

Completeness. Open completeness states that every o-normalizing term is typable, and with a tight derivation Φ such that $|\Phi|_{\mathsf{m}}$ is exactly the number of $\to_{\mathsf{om}}$ steps plus the open size of the o-normal form. The proof technique is standard: (*a*) a lemma states that every o-normal form is typable with a *tight* derivation; (*b*) subject expansion (Prop. 7.4) pulls back typability along $\to_{\mathsf{o}}$ steps; the exact bound is inherited from open correctness. A notable point is that, again, inert terms verify a special property: they can be given *any* multi type M.

Lemma 8.6 (Tight typability of open normal forms).

(1) Inert: *if t is an inert term then, for any multi type M, there is a type context Γ and a derivation $\Phi \rhd \Gamma \vdash t : M$; if, moreover, M is inert then Φ is inert.*
(2) Fireball: *if t is a fireball then there is a tight derivation $\Phi \rhd \Gamma \vdash t : \mathbf{0}$.*

Theorem 8.7 (Open completeness). *Let $d : t \to^*_{\mathsf{o}} u$ be an* o*-normalizing reduction sequence. Then there is a tight derivation $\Phi \rhd \Gamma \vdash t : \mathbf{0}$ such that $2|d|_{\mathsf{m}} + |u|_{\mathsf{o}} = |\Phi|_{\mathsf{m}}$.*

By the operational characterization of CbV scutability (Prop. 5.2.1), open completeness says that *every* VSC-scrutable term is typable with $\mathbf{0}$ and an inert type context.

9 MULTI TYPES FOR CBV SOLVABILITY

Here we provide both qualitative and quantitative characterizations of VSC solvable terms by studying the relationship between multi types and solving reduction $\to_{\mathsf{s}}$.

Solvable size. We need a notion of size for normal forms of solving reduction. The *solvable size* $|t|_{\mathsf{s}}$ of a term t is its number of applications plus its number of head abstractions.

$$|x|_{\mathsf{s}} := 0 \qquad |\lambda x.t|_{\mathsf{s}} := |t|_{\mathsf{s}} + 1 \qquad |tu|_{\mathsf{s}} := |t|_{\mathsf{s}} + |u|_{\mathsf{o}} + 1 \qquad |t[x \leftarrow u]|_{\mathsf{s}} := |t|_{\mathsf{s}} + |u|_{\mathsf{o}}.$$

Solvable Multi Types. The (qualitative) characterization of solvable terms with multi types is simple: they are those terms typable with a *solvable multi type*, defined in Figure 3. The idea is that an unsolvable term such as $t := \lambda x.\delta\delta$ should not be typable. It is typable only with $\mathbf{0}$, so we have to forbid the right-hand type to be $\mathbf{0}$. But then $\lambda y.t$, which is also unsolvable, is still typable, with *e.g.* $[\mathbf{0} \multimap \mathbf{0}]$. Now, the problem is the $\mathbf{0}$ on the *right* of $\multimap$, which is used to type t, and *not* the $\mathbf{0}$ on the left of $\multimap$, as it is needed to type solvable terms such as $\lambda y.x$ (which is typable with $[\mathbf{0} \multimap M]$ for any M). Therefore, solvable types forbids the right-hand type to be $\mathbf{0}$, and recursively to have $\mathbf{0}$ on the right of $\multimap$ inside the right-hand type. Such a constraint ultimately requires a ground multi type $n[X]$ different from $\mathbf{0}$ in the type system (in contrast to the open case, which does not need X).

Precisely Solvable Multi Types. Every solvable type derivation shall provide bounds, but for *exact* bounds two orthogonal predicates refining solvable types, namely *unitary solvable* and *inertly solvable types* (see Figure 3), are required.

The *unitary* predicate ensures that each solving multiplicative step is counted *exactly* once. Solvable types guarantee that each such step is counted, but it might be counted more than once. The constraint amounts to asking that the topmost and right-hand multisets are singletons. This is the key requirement for obtaining that in the statement of subject reduction the general size of the derivation decreases by exactly one at each multiplicative step.

The *inert* predicate (generalizing the one for the open case) ensures that the type derivation does not type subterms not accessible to solving reduction. The constraint is that the left-hand multisets have to be inert. As for the open case, the inert predicate enforces the matching of the size of solving normal forms with the size of their type derivation.

Solvable types that are both unitary and inert are called *precisely solvable*, and provide exact bounds, when the type context is also inert (to avoid typing the body of non-head abstractions).

Correctness. Solving correctness claims that solving reduction terminates for all terms typable with a solvable type M, and that the multiplicative size of a derivation bounds the number of $\to_{\mathsf{sm}}$ steps plus the solvable size of the s-normal form. This bound is exact if the type context is inert and M is precisely solvable. Modulo the new predicates, the proof follows the blueprint of the open case.

Lemma 9.1 (Size of solved fireballs). *Let f_{s} be a solved fireball and $\Phi \rhd \Gamma \vdash f_{\mathsf{s}} : M$.*

(1) Bounds*: if M is solvable then $|\Phi|_{\mathsf{m}} \geq |f_{\mathsf{s}}|_{\mathsf{s}}$.*

(2) Exact bounds*: if Γ is inert and M is precisely solvable then $|\Phi|_{\mathsf{m}} = |f_{\mathsf{s}}|_{\mathsf{s}}$.*

The only difference with the open case is that for quantitative solving subject reduction we also need the predicates. The sizes of type derivations decrease only if the right-hand type M is *solvable*, and decrease of the exact quantity only if M is *unitary* solvable. For the need for solvable types, consider the unsolvable term $\lambda y.\delta\delta$: it is typable only with $\mathbf{0}$ (which is *not* a solvable type) using a derivation Φ that does not type the body $\delta\delta$ of the abstraction (it is made of a many rule without premises). Its reduct, obtained by reducing the body, is still an abstraction, typable in the same way, and then the size of the derivation does not decrease.

Proposition 9.2 (Solving quantitative subject reduction). *Let $\Phi \rhd \Gamma \vdash t : M$ with M solvable.*

(1) Multiplicative step: *if $t \to_{\mathsf{sm}} t'$ then there is a derivation $\Phi' \rhd \Gamma \vdash t' : M$ such that $|\Phi'|_{\mathsf{m}} \leq |\Phi|_{\mathsf{m}} - 2$ and $|\Phi'| < |\Phi|$. If moreover M is unitary solvable then $|\Phi'|_{\mathsf{m}} = |\Phi|_{\mathsf{m}} - 2$ and $|\Phi'| = |\Phi| - 1$.*

(2) Exponential step: *if $t \to_{\mathsf{se}} t'$ then there is $\Phi' \rhd \Gamma \vdash t' : M$ with $|\Phi'|_{\mathsf{m}} = |\Phi|_{\mathsf{m}}$ and $|\Phi'| < |\Phi|$.*

Theorem 9.3 (Solving correctness). *Let $\Phi \rhd \Gamma \vdash t : M$ be a derivation with M solvable. Then, there is an* s*-normalizing reduction sequence $d : t \to_{\mathsf{s}}^* u$ with $2|d|_{\mathsf{m}} + |u|_{\mathsf{s}} \leq |\Phi|_{\mathsf{m}}$. If moreover Γ is a inert type context and M is precisely solvable then $2|d|_{\mathsf{m}} + |u|_{\mathsf{s}} = |\Phi|_{\mathsf{m}}$.*

Proof. By induction on the general size $|\Phi|$ of Φ.

If t is normal for $\to_{\mathsf{s}}$, then $t = f_{\mathsf{s}}$ is a solved fireball. Let d be the empty reduction sequence (so $|d|_{\mathsf{m}} = 0$), thus $|\Phi|_{\mathsf{m}} \geq |f_{\mathsf{s}}|_{\mathsf{s}} = |f_{\mathsf{s}}|_{\mathsf{s}} + 2|d|_{\mathsf{m}}$ (resp. $|\Phi|_{\mathsf{m}} = |f_{\mathsf{s}}|_{\mathsf{s}} = |f_{\mathsf{s}}|_{\mathsf{s}} + 2|d|_{\mathsf{m}}$) by Lemma 9.1.

Otherwise, t is not normal for $\to_{\mathsf{s}}$ and so $t \to_{\mathsf{s}} u$. According to solvable subject reduction (Prop. 9.2), there is a derivation $\Psi \rhd \Gamma \vdash u : M$ such that $|\Psi| < |\Phi|$ and

- $|\Psi|_{\mathsf{m}} \leq |\Phi|_{\mathsf{m}} - 2$ (resp. $|\Psi|_{\mathsf{m}} = |\Phi|_{\mathsf{m}} - 2$) if $t \to_{\mathsf{sm}} u$,
- $|\Psi|_{\mathsf{m}} = |\Phi|_{\mathsf{m}}$ if $t \to_{\mathsf{se}} u$.

By *i.h.*, there is a solved fireball f_{s} and a reduction sequence $d' : u \to_{\mathsf{s}}^* f_{\mathsf{s}}$ with $2|d'|_{\mathsf{m}} + |f_{\mathsf{s}}|_{\mathsf{s}} \leq |\Psi|_{\mathsf{m}}$ (resp. $2|d'|_{\mathsf{m}} + |f_{\mathsf{s}}|_{\mathsf{s}} = |\Psi|_{\mathsf{m}}$). Let d be the s-reduction sequence obtained by concatenating the first step $t \to_{\mathsf{s}} u$ and d'. There are two cases:

- *Multiplicative:* if $t \to_{\mathsf{sm}} u$ then $|\Phi|_{\mathsf{m}} \geq |\Psi|_{\mathsf{m}} + 2 \geq |f_{\mathsf{s}}|_{\mathsf{s}} + 2|d'|_{\mathsf{m}} + 2 = |f_{\mathsf{s}}|_{\mathsf{s}} + 2|d|_{\mathsf{m}}$ (resp. $|\Phi|_{\mathsf{m}} = |\Psi|_{\mathsf{m}} + 2 = |f_{\mathsf{s}}|_{\mathsf{s}} + 2|d'|_{\mathsf{m}} + 2 = |f_{\mathsf{s}}|_{\mathsf{s}} + 2|d|_{\mathsf{m}}$), since $|d|_{\mathsf{m}} = |d'|_{\mathsf{m}} + 1$.

- *Exponential:* if $t \to_{se} u$ then $|\Phi|_m = |\Psi|_m \geq |f_s|_s + 2|d'|_m = |f_s|_s + 2|d|_m$ (resp. $|\Phi|_m = |\Psi|_m = |f_s|_s + 2|d'|_m = |f_s|_s + 2|d|_m$), since $|d|_m = |d'|_m$. □

By the operational characterization of CbV solvability (Prop. 5.2.2), solving correctness says in particular that *only* VSC-solvable terms are typable with a solvable multi type.

Completeness. Solving completeness claims that every term such that its solving reduction terminates is typable with a precisely solvable type and an inert type context, and that the multiplicative size of the derivation is equal to the number of $\to_{sm}$ steps plus the solvable size of the s-normal form. Modulo the new predicates, the proof essentially follows the blueprint of the open case. In particular, completeness follows easily from the typability of solved fireballs.

Lemma 9.4 (Precisely solvable typability of solved fireballs). *If t is a solved fireball, then there is a derivation $\Phi \rhd \Gamma \vdash t : M$ with Γ inert type context and M precisely solvable.*

Theorem 9.5 (Solving completeness). *Let $d : t \to_s^* u$ be a* s*-normalizing reduction sequence. Then there is a derivation $\Phi \rhd \Gamma \vdash t : N$ with Γ inert, N precisely solvable and $2|d|_m + |u|_s = |\Phi|_m$.*

Proof. It suffices to prove that there is a derivation $\Phi \rhd \Gamma \vdash t : N$ with Γ inert and N precisely solvable. Indeed, by solvable correctness (Thm. 9.3), it follows then that there is an s-normalizing reduction sequence $d' : t \to_o^* u'$ such that $2|d'|_m + |u'|_s = |\Phi|_m$. By diamond and strong commutation (Prop. 3.4.2), $u' = u$ and $|d'|_m = |d|_m$. Let us prove that there is a derivation $\Phi \rhd \Gamma \vdash t : N$ with Γ inert and N precisely solvable, by induction on the length $|d|$ of the s-normalizing reduction $d : t \to_s^* u$.

If $|d| = 0$ then $|d|_m = 0$ and $t = u$ is s-normal and hence s_λ-normal. By Prop. 3.4.3, t is a solved fireball. By precisely solvable typability of solved fireballs (Lemma 9.4), there is a derivation $\Phi \rhd \Gamma \vdash t : N$ with Γ inert and N precisely solvable.

Otherwise, $|d| > 0$ and d is the concatenation of a first step $t \to_s s$ and a reduction sequence $d' : s \to_s^* u$, with $|d| = 1 + |d'|$. By *i.h.*, there is a derivation $\Psi \rhd \Gamma \vdash s : N$ with Γ inert and N precisely solvable. By subject expansion (Prop. 7.4, as $\to_s \subseteq \to_{vsc}$), there is a derivation $\Phi \rhd \Gamma \vdash t : N$. □

By the operational characterization of CbV solvability (Prop. 5.2.2), solving completeness says that *every* VSC-solvable term is typable with a precisely solvable type and an inert type context.

10 NORMALIZATION AND DENOTATIONAL SEMANTICS

Our study of multi types for Open CbV and CbV solvability also allows us to prove two normalization results: reductions $\to_o$ and $\to_s$ are *complete* with respect to their own normal forms, in the sense that if a term vsc-reduces to a o-normal (resp. s-normal) form, then reduction $\to_o$ (resp. $\to_s$) is enough to reach a—possibly different—o-normal (resp. s-normal) form. The proof exploits an elegant technique already used by de Carvalho et al. [2011] and Mazza et al. [2018].

Theorem 10.1 (Normalization). *Let t be a term in the* VSC.

(1) Open reduction: *if $t \to_{vsc}^* u$ where u is* o*-normal, then $t \to_o^* s$ for some* o*-normal s.*
(2) Solving reduction: *if $t \to_{vsc}^* u$ where u is* s*-normal, then $t \to_s^* s$ for some* s*-normal s.*

Proof. (1) Every o-normal form u is a fireball (Prop. 3.1.3) and hence has a derivation $\Phi \rhd \Gamma \vdash u : M$ (Lemma 8.6.2). Subject expansion (Prop. 7.4) iterated along $t \to_{vsc}^* u$ gives a derivation $\Psi \rhd \Gamma \vdash t : M$ for t. Open correctness (Thm. 8.5) gives $t \to_o^* s$ with s o-normal.

(2) Every s-normal form u is a solved fireball (Prop. 3.4.3) and hence has a derivation $\Phi \rhd \Gamma \vdash u : M$ (Lemma 9.4). Subject expansion (Prop. 7.4) iterated along $t \to_{vsc}^* u$ gives a derivation $\Psi \rhd \Gamma \vdash t : M$ for t. Solving correctness (Thm. 9.3) gives $t \to_s^* s$ with s s-normal. □

Thm. 10.1.1 is a generalization of the valuability result (Prop. 3.3.1) and it is the same as Prop. 3.3.2. Thm. 10.1.2 is the same as Prop. 3.5.1, but proved by type-theoretic means rather than operational.

Multi Types as Relational Semantics. Multi types induce a relational model[6] by interpreting a term as the set of its type judgments. Let t be a term and $x_1, \dots, x_n$ ($n \geq 0$) be pairwise distinct variables. The list $\vec{x} = (x_1, \dots, x_n)$ is *suitable for* t if $\mathrm{fv}(t) \subseteq \{x_1, \dots, x_n\}$. If $\vec{x} = (x_1, \dots, x_n)$ is suitable for t, the (*plain*) *semantics* $[\![t]\!]_{\vec{x}}$ *of* t *for* $\vec{x}$ and the *solvable semantics* $[\![t]\!]^{\mathrm{s}}_{\vec{x}}$ *of* t *for* $\vec{x}$ are defined by:

$$[\![t]\!]_{\vec{x}} := \{((N_1, \dots, N_n), M) \mid \exists\, \Phi \rhd x_1 : N_1, \dots, x_n : N_n \vdash t : M\}$$
$$[\![t]\!]^{\mathrm{s}}_{\vec{x}} := \{((N_1, \dots, N_n), M) \mid \exists \Phi \rhd x_1 : N_1, \dots, x_n : N_n \vdash t : M \text{ such that } M \text{ is solvable}\}.$$

Subject reduction and expansion (Prop. 7.4) guarantee that $[\![t]\!]_{\vec{x}}$ and $[\![t]\!]^{\mathrm{s}}_{\vec{x}}$ are *invariant* by $\to_{\mathrm{vsc}} \cup \equiv$. So, we provide two distinct *denotational* semantics not only for the (core) VSC, but also for its extension considered in Section 4, obtained by adding structural equivalence $\equiv$ to the core VSC.

Proposition 10.2 (Invariance). *Let t, u be terms in the* VSC *with $\vec{x} = (x_1, \dots, x_n)$ suitable for both of them. If t $(\to_{\mathrm{vsc}} \cup \equiv)$ u then $[\![t]\!]_{\vec{x}} = [\![u]\!]_{\vec{x}}$ and $[\![t]\!]^{\mathrm{s}}_{\vec{x}} = [\![u]\!]^{\mathrm{s}}_{\vec{x}}$.*

Open and solving correctness (Thms. 8.5 and 9.3) and completeness (Thms. 8.7 and 9.5) guarantee *adequacy* results for these semantics, *i.e.* a semantic characterization of CbV scrutability/solvability.

Theorem 10.3 (Adequacy). *Let t be a term in the* VSC *with $\vec{x} = (x_1, \dots, x_n)$ suitable for it.*

(1) Open: *$[\![t]\!]_{\vec{x}}$ is non-empty if and only if t is* o*-normalizing if and only if t is* VSC*-scrutable.*
(2) Solvable: *$[\![t]\!]^{\mathrm{s}}_{\vec{x}}$ is non-empty if and only if t* s*-normalizing if and only if t is* VSC*-solvable.*

Open adequacy (Thm. 10.3.1) implies that the equational theory $\mathcal{T}_{\mathrm{o}}$ induced by $[\![t]\!]_{\vec{x}}$ (equating all terms having the same semantics) is scrutable. The equational theory $\mathcal{T}_{\mathrm{s}}$ induced by $[\![t]\!]^{\mathrm{s}}_{\vec{x}}$ (equating all terms having the same solving semantics), instead, collapses all CbV unsolvable terms, and is thus *inconsistent* (Thm. 6.5). So—unlike $\mathcal{T}_{\mathrm{o}}$—the study of $\mathcal{T}_{\mathrm{s}}$ turns out to be pointless, although the solving semantics which induces that theory characterizes interesting operational properties.

Relational Semantics and CbV Models. Inspired by Hindley and Longo [1980], Egidi et al. [1992] proposed a set-theoretic and axiomatic definition of a CbV denotational model, later used and simplified by Ronchi Della Rocca et al. [2019; 1999; 2004]. Manzonetto et al. [2019] showed that a certain family of CbV multi type systems induce some CbV models (in the sense of Egidi et al. [1992]). Ehrhard's multi type system (Figure 2) used here does not belong to that family, but it shares the same philosophy based on two kinds of type, linear and multi. So, the proof in [Manzonetto et al. 2019] may be adapted to show that Ehrhard's multi type system induces a CbV model.

11 CONCLUSIONS

This paper shows that CbV solvability in the VSC has a rich theory, comparable to the one of CbN solvability in terms of characterizations, and yet different, as CbV unsolvable terms are not collapsible. A natural future direction is the refinement of behavioral equivalences such as Lassen's [2005] open CbV bisimilarity, which is not a scrutable theory: CbV inscrutable terms such as Ω, $(xy)\Omega$, and $(\lambda x.\delta)(yy)\delta$ (with $\delta := \lambda z.zz$ and $\Omega := \delta\delta$) are all distinct for his bisimilarity. At a more technical level, Ghilezan [2001] develops an interesting technique for proving the genericity lemma, based on a topology over λ-terms defined via intersection types. It would be interesting to see if it can be adapted to Ehrhard's multi types to prove genericity for CbV inscrutable terms.

ACKNOWLEDGMENTS

To Delia Kesner, Giulio Manzonetto and Xavier Montillet for many discussions on the topic. Also, we would have not submitted to ICFP if it were not for Delia's encouragement.

[6] Such a model is the restriction of the relational model for lineal logic to the image of Girard's [1987] CbV translation $(A \Rightarrow B)^{\mathrm{v}} = !(A^{\mathrm{v}} \multimap B^{\mathrm{v}})$ of the intuitionistic arrow into linear logic.

REFERENCES

Samson Abramsky. 1991. Domain Theory in Logical Form. *Ann. Pure Appl. Log.* 51, 1-2 (1991), 1–77. https://doi.org/10.1016/0168-0072(91)90065-T

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

Beniamino Accattoli, Andrea Condoluci, Giulio Guerrieri, and Claudio Sacerdoti Coen. 2019a. Crumbling Abstract Machines. In *Proceedings of the 21st International Symposium on Principles and Practice of Programming Languages, PPDP 2019, Porto, Portugal, October 7-9, 2019*. ACM, 4:1–4:15. https://doi.org/10.1145/3354166.3354169

Beniamino Accattoli, Andrea Condoluci, and Claudio Sacerdoti Coen. 2021a. Strong Call-by-Value is Reasonable, Implosively. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*. IEEE, 1–14. https://doi.org/10.1109/LICS52264.2021.9470630

Beniamino Accattoli and Ugo Dal Lago. 2012. On the Invariance of the Unitary Cost Model for Head Reduction. In *23rd International Conference on Rewriting Techniques and Applications (RTA'12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 22–37. https://doi.org/10.4230/LIPIcs.RTA.2012.22

Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. 2021b. The (In)Efficiency of interaction. *Proc. ACM Program. Lang.* 5, POPL (2021), 1–33. https://doi.org/10.1145/3434332

Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. 2021c. The Space of Interaction. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*. IEEE, 1–13. https://doi.org/10.1109/LICS52264.2021.9470726

Beniamino Accattoli, Stéphane Graham-Lengrand, and Delia Kesner. 2018. Tight typings and split bounds. *PACMPL* 2, ICFP (2018), 94:1–94:30. https://doi.org/10.1145/3236789

Beniamino Accattoli and Giulio Guerrieri. 2016. Open Call-by-Value. In *Programming Languages and Systems - 14th Asian Symposium, APLAS 2016, Hanoi, Vietnam, November 21-23, 2016, Proceedings (Lecture Notes in Computer Science, Vol. 10017)*. Springer, 206–226. https://doi.org/10.1007/978-3-319-47958-3_12

Beniamino Accattoli and Giulio Guerrieri. 2018. Types of Fireballs. In *Programming Languages and Systems - 16th Asian Symposium, APLAS 2018, Wellington, New Zealand, December 2-6, 2018, Proceedings (Lecture Notes in Computer Science, Vol. 11275)*. Springer, 45–66. https://doi.org/10.1007/978-3-030-02768-1_3

Beniamino Accattoli and Giulio Guerrieri. 2022. The Theory of Call-by-Value Solvability (long version). *CoRR* abs/2207.08697 (2022). https://arxiv.org/abs/2207.08697

Beniamino Accattoli, Giulio Guerrieri, and Maico Leberle. 2019b. Types by Need. In *Programming Languages and Systems - 28th European Symposium on Programming, ESOP 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 (Lecture Notes in Computer Science, Vol. 11423)*. Springer, 410–439. https://doi.org/10.1007/978-3-030-17184-1_15

Beniamino Accattoli, Giulio Guerrieri, and Maico Leberle. 2021d. Semantic Bounds and Strong Call-by-Value Normalization. *CoRR* abs/2104.13979 (2021). https://arxiv.org/abs/2104.13979

Beniamino Accattoli and Luca Paolini. 2012. Call-by-Value Solvability, Revisited. In *Functional and Logic Programming - 11th International Symposium, FLOPS 2012, Kobe, Japan, May 23-25, 2012. Proceedings*. Springer, 4–16. https://doi.org/10.1007/978-3-642-29822-6_4

Beniamino Accattoli and Claudio Sacerdoti Coen. 2015. On the Relative Usefulness of Fireballs. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*. IEEE, 141–155. https://doi.org/10.1109/LICS.2015.23

Beniamino Accattoli and Claudio Sacerdoti Coen. 2017. On the value of variables. *Information and Computation* 255 (2017), 224–242. https://doi.org/10.1016/j.ic.2017.01.003

Sandra Alves, Delia Kesner, and Daniel Ventura. 2019. A Quantitative Understanding of Pattern Matching. In *25th International Conference on Types for Proofs and Programs, TYPES 2019, June 11-14, 2019, Oslo, Norway (LIPIcs, Vol. 175)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 3:1–3:36. https://doi.org/10.4230/LIPIcs.TYPES.2019.3

Henk Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. 1983. A Filter Lambda Model and the Completeness of Type Assignment. *J. Symb. Log.* 48, 4 (1983), 931–940. https://doi.org/10.2307/2273659

Hendrik Pieter Barendregt. 1971. *Some extensional term models for combinatory logics and λ-calculi*. Ph. D. Dissertation. Univ. Utrecht.

Hendrik Pieter Barendregt. 1977. Solvability in lambda-calculi. In *Colloque international de logique : Clermont-Ferrand, 18-25 juillet 1975*, M. Guillaume (Ed.). Éditions du C.N.R.S., Paris, 209–219.

Hendrik Pieter Barendregt. 1984. *The Lambda Calculus – Its Syntax and Semantics*. Vol. 103. North-Holland.

Alexis Bernadet and Stéphane Lengrand. 2013. Non-idempotent intersection types and strong normalisation. *Logical Methods in Computer Science* 9, 4 (2013). https://doi.org/10.2168/LMCS-9(4:3)2013

Antonio Bucciarelli and Thomas Ehrhard. 2001. On phase semantics and denotational semantics: the exponentials. *Ann. Pure Appl. Logic* 109, 3 (2001), 205–241. https://doi.org/10.1016/S0168-0072(00)00056-7

Antonio Bucciarelli, Delia Kesner, Alejandro Ríos, and Andrés Viso. 2020. The Bang Calculus Revisited. In *Functional and Logic Programming - 15th International Symposium, FLOPS 2020, Akita, Japan, September 14-16, 2020, Proceedings*. Springer, 13–32. https://doi.org/10.1007/978-3-030-59025-3_2

Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. 2021. Solvability = Typability + Inhabitation. *Log. Methods Comput. Sci.* 17, 1 (2021). https://doi.org/10.23638/LMCS-17(1:7)2021

Antonio Bucciarelli, Delia Kesner, and Daniel Ventura. 2017. Non-idempotent intersection types for the Lambda-Calculus. *Logic Journal of the IGPL* 25, 4 (2017), 431–464. https://doi.org/10.1093/jigpal/jzx018

Alberto Carraro and Giulio Guerrieri. 2014. A Semantical and Operational Account of Call-by-Value Solvability. In *Foundations of Software Science and Computation Structures - 17th International Conference, FOSSACS 2014, Grenoble, France, April 5-13, 2014, Proceedings*. Springer, 103–118. https://doi.org/10.1007/978-3-642-54830-7_7

Mario Coppo and Mariangiola Dezani-Ciancaglini. 1978. A new type assignment for λ-terms. *Arch. Math. Log.* 19, 1 (1978), 139–156. https://doi.org/10.1007/BF02011875

Mario Coppo and Mariangiola Dezani-Ciancaglini. 1980. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic* 21, 4 (1980), 685–693. https://doi.org/10.1305/ndjfl/1093883253

Mario Coppo, Mariangiola Dezani-Ciancaglini, and Maddalena Zacchi. 1987. Type Theories, Normal Forms and D_∞-Lambda-Models. *Inf. Comput.* 72, 2 (1987), 85–116. https://doi.org/10.1016/0890-5401(87)90042-3

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

Ugo Dal Lago, Claudia Faggian, and Simona Ronchi Della Rocca. 2021. Intersection types and (positive) almost-sure termination. *Proc. ACM Program. Lang.* 5, POPL (2021), 1–32. https://doi.org/10.1145/3434313

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

Daniel de Carvalho. 2018. Execution time of λ-terms via denotational semantics and intersection types. *Math. Str. in Comput. Sci.* 28, 7 (2018), 1169–1203. https://doi.org/10.1017/S0960129516000396

Daniel de Carvalho, Michele Pagani, and Lorenzo Tortora de Falco. 2011. A semantic measure of the execution time in linear logic. *Theor. Comput. Sci.* 412, 20 (2011), 1884–1902. https://doi.org/10.1016/j.tcs.2010.12.017

Daniel de Carvalho and Lorenzo Tortora de Falco. 2016. A semantic account of strong normalization in linear logic. *Inf. Comput.* 248 (2016), 104–129. https://doi.org/10.1016/j.ic.2015.12.010

Roy Dyckhoff and Stéphane Lengrand. 2007. Call-by-Value lambda-calculus and LJQ. *J. Log. Comput.* 17, 6 (2007), 1109–1134. https://doi.org/10.1093/logcom/exm037

Lavinia Egidi, Furio Honsell, and Simona Ronchi Della Rocca. 1992. Operational, denotational and logical descriptions: a case study. *Fundam. Inform.* 16, 1 (1992), 149–169.

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

José Espírito Santo. 2020. The Call-By-Value Lambda-Calculus with Generalized Applications. In *28th EACSL Annual Conference on Computer Science Logic, CSL 2020, January 13-16, 2020, Barcelona, Spain (LIPIcs, Vol. 152)*, Maribel Fernández and Anca Muscholl (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 35:1–35:12. https://doi.org/10.4230/LIPIcs.CSL.2020.35

Álvaro García-Pérez and Pablo Nogueira. 2016. No solvable lambda-value term left behind. *Logical Methods in Computer Science* 12, 2 (2016). https://doi.org/10.2168/LMCS-12(2:12)2016

Philippa Gardner. 1994. Discovering Needed Reductions Using Type Theory. In *TACS '94 (Lecture Notes in Computer Science, Vol. 789)*. Springer, 555–574. https://doi.org/10.1007/3-540-57887-0_115

Silvia Ghilezan. 2001. Full Intersection Types and Topologies in Lambda Calculus. *J. Comput. Syst. Sci.* 62, 1 (2001), 1–14. https://doi.org/10.1006/jcss.2000.1703

Jean-Yves Girard. 1987. Linear Logic. *Theoretical Computer Science* 50 (1987), 1–102. https://doi.org/10.1016/0304-3975(87)90045-4

Jean-Yves Girard. 1988. Normal functors, power series and the λ-calculus. *Annals of Pure and Applied Logic* 37 (1988), 129–177. https://doi.org/10.1016/0168-0072(88)90025-5

Benjamin Grégoire and Xavier Leroy. 2002. A compiled implementation of strong reduction. In *Proceedings of the Seventh ACM SIGPLAN International Conference on Functional Programming, ICFP '02*. ACM, 235–246. https://doi.org/10.1145/581478.581501

Giulio Guerrieri. 2015. Head reduction and normalization in a call-by-value lambda-calculus. In *2nd International Workshop on Rewriting Techniques for Program Transformations and Evaluation, WPTE 2015, July 2, 2015, Warsaw, Poland (OASIcs, Vol. 46)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 3–17. https://doi.org/10.4230/OASIcs.WPTE.2015.3

Giulio Guerrieri. 2019. Towards a Semantic Measure of the Execution Time in Call-by-Value lambda-Calculus. In *Proceedings Twelfth Workshop on Developments in Computational Models and Ninth Workshop on Intersection Types and Related Systems, DCM/ITRS 2018. (EPTCS, Vol. 293)*. 57–72. https://doi.org/10.4204/EPTCS.293.5

Giulio Guerrieri, Luca Paolini, and Simona Ronchi Della Rocca. 2015. Standardization of a Call-By-Value Lambda-Calculus. In *13th International Conference on Typed Lambda Calculi and Applications, TLCA 2015, July 1-3, 2015, Warsaw, Poland (LIPIcs, Vol. 38)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 211–225. https://doi.org/10.4230/LIPIcs.TLCA.2015.211

Giulio Guerrieri, Luca Paolini, and Simona Ronchi Della Rocca. 2017. Standardization and Conservativity of a Refined Call-by-Value lambda-Calculus. *Logical Methods in Computer Science* 13, 4 (2017). https://doi.org/10.23638/LMCS-13(4:29)2017

Hugo Herbelin and Stéphane Zimmermann. 2009. An operational account of Call-by-Value Minimal and Classical λ-calculus in Natural Deduction form. In *Typed Lambda Calculi and Applications, 9th International Conference, TLCA 2009, Brasilia, Brazil, July 1-3, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5608)*. Springer, 142–156. https://doi.org/10.1007/978-3-642-02273-9_12

Roger Hindley and Giuseppe Longo. 1980. Lambda-Calculus Models and Extensionality. *Mathematical Logic Quarterly* 26, 19-21 (1980), 289–310. https://doi.org/10.1002/malq.19800261902

Furio Honsell and Simona Ronchi Della Rocca. 1992. An Approximation Theorem for Topological Lambda Models and the Topological Incompleteness of Lambda Calculus. *J. Comput. Syst. Sci.* 45, 1 (1992), 49–75. https://doi.org/10.1016/0022-0000(92)90040-P

Richard Kennaway, Vincent van Oostrom, and Fer-Jan de Vries. 1999. Meaningless Terms in Rewriting. *J. Funct. Log. Program.* 1999, 1 (1999).

Axel Kerinec, Giulio Manzonetto, and Simona Ronchi Della Rocca. 2021. Call-By-Value, Again!. In *6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021 (LIPIcs, Vol. 195)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 7:1–7:18. https://doi.org/10.4230/LIPIcs.FSCD.2021.7

Delia Kesner, Loïc Peyrot, and Daniel Ventura. 2021. The Spirit of Node Replication. In *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12650)*. Springer, 344–364. https://doi.org/10.1007/978-3-030-71995-1_18

Delia Kesner and Pierre Vial. 2020. Consuming and Persistent Types for Classical Logic. In *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*. 619–632. https://doi.org/10.1145/3373718.3394774

Delia Kesner and Andrés Viso. 2022. Encoding Tight Typing in a Unified Framework. In *28th EACSL Annual Conference on Computer Science Logic, CSL 2020, January 13-16, 2020, Barcelona, Spain (LIPIcs, Vol. 216)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 27:1–27:20. https://doi.org/10.4230/LIPIcs.CSL.2022.27

Assaf J. Kfoury. 2000. A linearization of the Lambda-calculus and consequences. *J. Log. Comput.* 10, 3 (2000), 411–436. https://doi.org/10.1093/logcom/10.3.411

Jean-Louis Krivine. 1993. *Lambda-calculus, types and models*. Masson.

Søren B. Lassen. 2005. Eager Normal Form Bisimulation. In *20th IEEE Symposium on Logic in Computer Scienc, LICS 2005*. IEEE Computer Society, 345–354. https://doi.org/10.1109/LICS.2005.15

Paul Blain Levy, John Power, and Hayo Thielecke. 2003. Modelling environments in call-by-value programming languages. *Inf. Comput.* 185, 2 (2003), 182–210. https://doi.org/10.1016/S0890-5401(03)00088-9

Giuseppe Longo. 1983. Set-theoretical models of λ-calculus: theories, expansions, isomorphisms. *Ann. Pure Appl. Log.* 24, 2 (1983), 153–188. https://doi.org/10.1016/0168-0072(83)90030-1

Giulio Manzonetto, Michele Pagani, and Simona Ronchi Della Rocca. 2019. New Semantical Insights Into Call-by-Value λ-Calculus. *Fundam. Inform.* 170, 1-3 (2019), 241–265. https://doi.org/10.3233/FI-2019-1862

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

Damiano Mazza, Luc Pellissier, and Pierre Vial. 2018. Polyadic Approximations, Fibrations and Intersection Types. *Proceedings of the ACM on Programming Languages* 2, POPL (2018), 6:1–6:28. https://doi.org/10.1145/3158094

Eugenio Moggi. 1988. *Computational λ-Calculus and Monads*. LFCS report ECS-LFCS-88-66. University of Edinburgh. http://www.lfcs.inf.ed.ac.uk/reports/88/ECS-LFCS-88-66/ECS-LFCS-88-66.pdf

Eugenio Moggi. 1989. Computational λ-Calculus and Monads. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989*. IEEE Computer Society, 14–23. https://doi.org/10.1109/LICS.1989.39155

Peter Møller Neergaard and Harry G. Mairson. 2004. Types, potency, and idempotency: why nonlinearity and amnesia make a type system work. In *ICFP 2004*. 138–149. https://doi.org/10.1145/1016850.1016871

Luca Paolini. 2001. Call-by-Value Separability and Computability. In *Theoretical Computer Science, 7th Italian Conference, ICTCS 2001, Torino, Italy, October 4-6, 2001, Proceedings*. 74–89. https://doi.org/10.1007/3-540-45446-2_5

Luca Paolini and Simona Ronchi Della Rocca. 1999. Call-by-value Solvability. *RAIRO Theor. Informatics Appl.* 33, 6 (1999), 507–534. https://doi.org/10.1051/ita:1999130

Andrew M. Pitts. 2012. Howe's method for higher-order languages. In *Advanced Topics in Bisimulation and Coinduction*, Davide Sangiorgi and Jan J. M. M. Rutten (Eds.). Cambridge tracts in theoretical computer science, Vol. 52. Cambridge University Press, 197–232.

Gordon D. Plotkin. 1975. Call-by-Name, Call-by-Value and the lambda-Calculus. *Theoretical Computer Science* 1, 2 (1975), 125–159. https://doi.org/10.1016/0304-3975(75)90017-1

Gordon D. Plotkin. 1993. Set-Theoretical and Other Elementary Models of the lambda-Calculus. *Theor. Comput. Sci.* 121, 1&2 (1993), 351–409. https://doi.org/10.1016/0304-3975(93)90094-A

Garrel Pottinger. 1980. A type assignment for the strongly normalizable λ-terms. In *To HB Curry: essays on combinatory logic, λ-calculus and formalism*. 561–577.

Alberto Pravato, Simona Ronchi Della Rocca, and Luca Roversi. 1999. The call-by-value λ-calculus: a semantic investigation. *Math. Str. in Comput. Sci.* 9, 5 (1999), 617–650.

Simona Ronchi Della Rocca and Luca Paolini. 2004. *The Parametric λ-Calculus – A Metamodel for Computation*. Springer. https://doi.org/10.1007/978-3-662-10394-4

Amr Sabry and Matthias Felleisen. 1993. Reasoning about Programs in Continuation-Passing Style. *Lisp and Symbolic Computation* 6, 3-4 (1993), 289–360.

Amr Sabry and Philip Wadler. 1997. A Reflection on Call-by-Value. *ACM Trans. Program. Lang. Syst.* 19, 6 (1997), 916–941. https://doi.org/10.1145/267959.269968

Masako Takahashi. 1994. A Simple Proof of the Genericity Lemma. In *Logic, Language and Computation (Lecture Notes in Computer Science, Vol. 792)*. Springer, 117–118. https://doi.org/10.1007/BFb0032397

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

Christopher P. Wadsworth. 1976. The Relation Between Computational and Denotational Properties for Scott's D_∞-Models of the Lambda-Calculus. *SIAM J. Comput.* 5, 3 (1976), 488–521. https://doi.org/10.1137/0205036