Meaningfulness and Genericity in a Subsuming Framework

Delia Kesner
Université Paris Cité – CNRS – IRIF, France

Victor Arrial
Université Paris Cité – CNRS – IRIF, France

Giulio Guerrieri
University of Sussex, Department of Informatics, Brighton, United Kingdom

Abstract

This paper studies the notion of meaningfulness for a unifying framework called `dBang`-calculus, which subsumes both call-by-name (`dCBN`) and call-by-value (`dCBV`). We first define meaningfulness in `dBang` and then characterize it by means of typability and inhabitation in an associated non-idempotent intersection type system previously appearing in the literature. We validate the proposed notion of meaningfulness by showing two properties: (1) consistency of the smallest theory, called $\mathcal{H}$, equating all meaningless terms, and (2) genericity, stating that meaningless subterms have no bearing on the significance of meaningful terms. The theory $\mathcal{H}$ is also shown to have a unique consistent and maximal extension $\mathcal{H}^*$, which coincides with a well-known notion of observational equivalence. Last but not least, we show that the notions of meaningfulness and genericity in the literature for `dCBN` and `dCBV` are subsumed by the corresponding ones proposed here for the `dBang`-calculus.

2012 ACM Subject Classification Theory of computation → Operational semantics

Keywords and phrases Lambda calculus, Solvability, Meaningfulness, Inhabitation, Genericity

Digital Object Identifier 10.4230/LIPIcs.FSCD.2024.1

Category Invited Talk

Related Version *Full Version*: `https://arxiv.org/abs/2404.06361` [50]

1 Introduction

A common line of research in logic and theoretical computer science is to find unifying frameworks that subsume different paradigms, systems or calculi. Examples are call-by-push-value [54, 55], polarized system LU [45], linear calculi [57, 58, 72], bang-calculus [38, 39, 23, 24], system L [62, 35], ecumenical systems [68], monadic calculus [60, 61], and others [71, 40, 73].

The relevance of these unifying frameworks lies in the range of properties and models they encompass. Finding *unifying and simple primitives, tools and techniques* to reason about properties of different systems is challenging, and provides a deeper and more abstract understanding of these properties. The advantages of this kind of approach are numerous, for instance the *several-for-one deal*: study a property in a unifying framework gives appropriate intuitions and hints for free for all the subsumed systems. The aim of this paper is to go beyond the state of the art in a framework subsuming the *call-by-name* and *call-by-value* evaluation mechanisms, by unifying their notions of *meaningful* (and *meaningless*) programs.

Call-by-name and call-by-value. Every programming language implements a particular evaluation strategy, specifying when and how parameters are evaluated during function calls. For example, in call-by-value (`CBV`), the argument is evaluated before being passed to the function, while in call-by-name (`CBN`) the argument is passed immediately to the function

9th International Conference on Formal Structures for Computation and Deduction (FSCD 2024).
Editor: Jakob Rehof; Article No. 1; pp. 1:1–1:24
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

body, so that it may never be evaluated, or may be re-evaluated several times. These models of computation serve as the basis for many theoretical and practical studies in programming languages and proof assistants, such as OCaml, Haskell, Coq, Isabelle, etc.

The `CBN` strategy has garnered significant attention in the literature on theoretical studies and is generally perceived as well-established. In contrast, the `CBV` strategy has received limited attention. Despite their similarities, `CBN` and `CBV` strategies have predominantly been studied independently, leading to a fragmented research. This approach not only duplicates research efforts – once for `CBN` and once again for `CBV` – but also generally results in ad-hoc methods for dealing with the `CBV` case that are naively adapted from the `CBN` one.

Understanding the (logical) duality between `CBN` and `CBV` (*e.g.* [34]) marked a significant step towards properly unifying these models. It paved the way for the emergence of Call-by-Push-Value (`CBPV`), a unifying framework introduced by P.B. Levy [54, 55] which *subsumes*, among others, `CBN` and `CBV` denotational and operational semantics thanks to the distinction between *computations* and *values*, according to the slogan "a value is, a computation does". This framework attracts growing attention: proving advanced properties of a single *unifying paradigm*, and subsequently instantiate them for a *wide range* of computational models.

The distant `Bang`-calculus. Drawing inspiration from Girard's Linear Logic (LL) [44] and the interpretation of `CBPV` into LL [38], Ehrhard and Guerrieri [39] introduced an (untyped) restriction of `CBPV`, named `Bang`-*calculus*, already capable of subsuming both `CBN` and `CBV`. It is obtained by enriching the λ-calculus with two modalities ! and its dual `der`. The modality ! actually plays a twofold role: it freezes the evaluation of subterms (called *thunk* in `CBPV`), and it marks what can be duplicated or erased during evaluation (*i.e.* copied an arbitrary number of times, including zero). The modality `der` annihilates the effect of !, effectively restoring computation and eliminating duplicability. Embedding `CBN` or `CBV` into the `Bang`-calculus via *Girard's translations* simply consists in decorating λ-terms with ! and `der`, thereby forcing one model of computation or the other one. Thanks to these elementary modalities and embeddings, the `Bang`-calculus eases the identification of shared behaviors and properties of `CBN` and `CBV`, encompassing both syntactic and semantic aspects of them.

The original `Bang`-calculus [39] uses some permutation rules, similar to the ones used in [70, 30], that unveil hidden redexes and unblock reductions that otherwise would be stuck. These permutation rules make the calculus *adequate*, preventing some normal forms from being observationally equivalent to non-terminating terms. A major drawback is that the resulting combined reduction is not confluent (Page 6 in [39]). The *distant* `Bang`-*calculus* (`dBang`) [23, 24] was proposed as an adequate *and* confluent alternative. This is achieved by enriching the syntax with *explicit substitutions*, in the vein of Accattoli and Kesner's linear substitution calculus [7, 9, 1, 2] (generalizing in turn Milner's calculus [59, 51]), thanks to rewrite rules that act *at a distance*, so that permutation rules are no longer needed.

In this paper, we focus on `dBang`, and its relations with `dCBN` [9, 1] and `dCBV` [11], which are *distant* adequate variants of the `CBN` and `CBV` λ-calculi. This unifying framework is fruitful, subsuming numerous `dCBN` and `dCBV` properties through their associated embedding, as for instance big step semantics: evaluating the result from the `dCBN`/`dCBV` embedding of a given program t with the `dBang` model actually corresponds to the embedding of the result of evaluating the original program t with the `dCBN`/`dCBV` model. In other words, `dBang` is a language that breaks down the `dCBN` and `dCBV` paradigms into elementary primitives.

Let us now review the state of the art by discussing some advanced properties of programming languages that have been studied in the literature by using the unifying approach `dBang`. Some of these results, including this work, strongly rely on semantical tools such as quantitative types. To ensure clarity regarding the state of the art, let us briefly discuss in first place the main ideas behind quantitative types.

Quantitative Type Systems. *Intersection type systems* [31, 32] increase the typability power on λ-terms with respect to simple types by introducing a new *intersection* type constructor $\wedge$ that is associative, commutative and *idempotent* (*i.e.* $\sigma \wedge \sigma = \sigma$). Intersection types allow terms to have different types simultaneously, *e.g.* a term has type $\sigma \wedge \tau$ whenever it has both types σ and τ. They constitute a powerful tool to reason about *qualitative* properties of programs. For example, different notions of normalization can be characterized using intersection types [67, 33], in that a term t is typable in a given system if and only if t is normalizing (as a consequence, typability in these systems is undecidable). An alternative version of intersection type systems for the λ-calculus, called *non-idempotent* [43, 36], is obtained by dropping idempotence. In such a setting, a term of type $\sigma \wedge \sigma \wedge \tau$ can be seen as a resource used exactly once as a data of type τ and twice as a data of type σ. Interestingly, such type systems provide not only qualitative characterizations of different operational properties, but also *quantitative* ones: *e.g.* a term t is still typable if and only if t is normalizing, moreover any type derivation of t gives an *upper bound* to the execution time for t (the number of steps to reach a normal form) [37]. These upper bounds can be further refined into *exact measure* using *tight non-idempotent typing systems*, as pioneered in [4].

State of the Art. This paper contributes to a broader initiative aimed at consolidating the theory of `dCBN` and `dCBV`, by unifying them into `dBang`. Several results have already been factorized and generalized in this framework, we now revisit some of them.

In [46], it is shown that the interpretation of a term t in any denotational model of `CBN`/`CBV` obtained from LL is included in the interpretation of the `CBN`/`CBV` translation of t in any denotational model of `Bang` obtained from LL. The reverse inclusion also holds for `CBN` but not for `CBV`. In particular, these results apply to *typability* in non-idempotent intersection type systems inspired by LL. Indeed, typing is preserved by Girard's translations, meaning that if a term is typable in the `CBN`/`CBV` type system, then its `CBN`/`CBV` translation is typable in the type system $\mathcal{B}$ for `Bang`, using the same types. The converse holds for `CBN` but not for `CBV`. In [23, 24], the `CBV` typing system is modified so that the reverse implication also holds. Moreover, an extension of Girard's `CBN` translation to `dCBN` and a *new* `CBV` translation for `dCBV` are proposed. Similar typing preservation results have been obtained in [52] for the translations in [23, 24], but for the more precise notion of tight typing introduced in [4].

Retrieving *dynamic* properties from `Bang` into `CBN` and `CBV` turns out to be a more intricate task, especially in their *adequate* (distant) variant [23, 41, 24].

In [46] it is shown that `CBN` and `CBV` can be simulated by *reduction* in `Bang` through Girard's original translations. But the `CBV` translation fails to preserve *normal forms*, as some `CBV` normal forms translate to reducible terms in `Bang`. This issue is solved in `dBang` [23, 24], thanks to the *new* `CBV` translation for `dCBV` previously mentioned. In the end, reductions *and* normal forms are preserved by both the `CBN` and the new `CBV` translations.

Even if `dCBN` and `dCBV` can be both simulated by *reduction* in `dBang`, the converse, known as *reverse simulation*, holds for `dCBN` but fails for `dCBV` [24, 14]: a `dBang` reduction sequence from a term in the image of the `dCBV` embedding may not correspond to a valid reduction sequence in `dCBV`. Yet another new `dCBV` translation is proposed in [14] so that simulation and reverse simulation are now recovered.

Another major contribution concerns the *inhabitation* problem: given an environment Γ (a type assignment for variables) and a type σ, decide whether there is a term t that can be typed with σ under the environment Γ. While inhabitation was shown [74] to be *undecidable* in `CBN` for idempotent intersection type systems, it turns out to be *decidable* [25, 28] in the non-idempotent setting. Decidability of the inhabitation problem leads to the development of automatic tools for type-based *program synthesis* [56, 21], whose goal is to construct a program – the term t – that satisfies some high-level formal specification, expressed as a type σ with some assumptions described by the environment Γ. It has been proved in [13] that the algorithms deciding the inhabitation problem for dCBN and dCBV can be inferred from the corresponding one for dBang, thus providing a unified solution to this relevant problem.

Meaningfulness and Genericity. In this work, we aim to unify the notions of meaningfulness and genericity in dCBN and dCBV so as to derive them from the respective ones in dBang.

A naive approach to set a semantics for the pure untyped λ-calculus is to define the meaning of a β-normalizing λ-term as its normal form, and equating all λ-terms that do not β-normalize. The underlying idea is that, as β-reduction represents evaluation and a normal form stands for its outcome, all non-β-normalizing λ-terms (*i.e.* diverging programs) are then considered as meaningless. However, this simplistic approach is flawed, as thoroughly discussed in [20]. For example, any λ-theory equating all non-β-normalizing λ-terms is inherently inconsistent – it effectively equates all λ-terms, not just the meaningless ones!

Alternatively, during the 70s, Wadsworth [75, 76] and Barendregt [17, 18, 19, 20] showed that the meaningful (CBN) λ-terms can be identified with the *solvable* ones. Solvability is defined in a rather technical way: a λ-term t is *solvable* if there is a special kind of context, called *head* context $\mathtt{H}$, sending t to the identity function $\mathtt{I} = \lambda z.z$, meaning that $\mathtt{H}\langle t\rangle$ β-reduces to $\mathtt{I}$. Roughly, a solvable λ-term t may be divergent, but its diverging subterms can be eliminated by supplying the right arguments to t via an appropriate interaction with a suitable head context $\mathtt{H}$. For instance, in CBN, $x\Omega$ is divergent but solvable using the head context $\mathtt{H} = (\lambda x.\diamond)(\lambda y.\mathtt{I})$. It turns out that *unsolvable* λ-terms constitutes a strict subset of the non-β-normalizing ones. Moreover, the smallest λ-theory that equates all unsolvable λ-terms is *consistent* (*i.e.* it does not equate all terms). In Barendregt's book [20], these results rely on a keystone property known as (*full*) *genericity*, which states that meaningless subterms are computationally irrelevant – in the sense that they do not play any role – in the evaluation of β-normalizing terms. Formally, if t is *unsolvable* and $\mathtt{C}\langle t\rangle$ β-reduces to *some* β-normal term u for some context $\mathtt{C}$, then $\mathtt{C}\langle s\rangle$ β-reduces to u for *every* λ-term s. This property stands as a fool guard that the choice of meaningfulness is adequate. A variant of genericity [16], called *surface* in [15] and *light* in [10], states that any meaningless subterm t is irrelevant in a meaningful term $\mathtt{C}\langle t\rangle$ in that $\mathtt{C}\langle s\rangle$ is still meaningful, for every term u.

Meaningfulness was also studied for first order rewriting systems [48] and other strategies of the λ-calculus [71]. Notably, finding the correct notion of meaningfulness for CBV has been a challenge [5, 6, 15]. Similarly, an extension of the dCBN was studied [29, 26] in the framework of a λ-calculus equipped with pattern matching for pairs. The use of different data structures in the language – functions and pairs – makes meaningfulness more challenging. Indeed, it was shown that meaningfulness cannot be characterized only by means of typability alone, as in CBN and CBV, but also requires some additional conditions stated in terms of the inhabitation problem previously mentioned. This result for the λ-calculus with patterns inspired the characterization of meaningfulness for dBang that we provide in this paper. Genericity for dCBN and the more subtle case of dCBV was recently proved in [15].

Our Contributions. We first define meaningfulness for dBang, for which we provide a characterization by means of typability *and* inhabitation. As a second contribution, we validate this notion of meaningfulness twofold: meaningless terms enjoy surface genericity, and the smallest $\lambda_{\mathtt{dBang}}$-theory $\mathcal{H}_{\mathtt{dBang}}$ obtained by equating all the meaningless terms is consistent. Moreover, we show that $\mathcal{H}_{\mathtt{dBang}}$ admits a unique maximal consistent extension

$\mathcal{H}^*_{\mathtt{dBang}}$ and show that it coincides with the well-known notion of observational equivalence. Last but not least, as a third contribution, we show that the notions of meaningfulness in the literature for dCBN and dCBV are subsumed by the one proposed here for dBang. We also obtain surface genericity for dCBN and dCBV as a consequence of the genericity property for dBang, and relate the theories $\mathcal{H}_{\mathtt{dBang}}$ and $\mathcal{H}^*_{\mathtt{dBang}}$ (in dBang) to the corresponding ones in dCBN and dCBV. Detailed proofs of our results can be found in [50].

Roadmap. Section 2 recalls dBang and its quantitative type system $\mathcal{B}$. Section 3 defines meaningfulness for dBang, and characterizes it in terms of typability and inhabitation in the type system $\mathcal{B}$. Section 4 addresses surface genericity and the construction of the theories $\mathcal{H}_{\mathtt{dBang}}$ and $\mathcal{H}^*_{\mathtt{dBang}}$, while Section 5 establishes a precise relationship between meaningless and genericity in dCBN/dCBV and their corresponding notions in dBang. Section 6 discusses future and related work and concludes.

2 The dBang-Calculus

2.1 Syntax and Operational Semantics

We introduce the syntax of the *distant* Bang-*calculus* (dBang) [23, 24]. Given a countably infinite set $\mathcal{X}$ of variables $x, y, z, \ldots$, the set $\Lambda_!$ of *terms* is inductively defined as follows:

$$\textbf{(Terms)} \quad t, u, s \coloneqq x \in \mathcal{X} \mid t\,u \mid \lambda x.t \mid t[x\backslash u] \mid !t \mid \mathtt{der}(t)$$

The set $\Lambda_!$ includes **variables** x, **abstractions** $\lambda x.t$ and **applications** tu (as in the λ-calculus), and three other constructors: a **closure** $t[x\backslash u]$ representing a pending **explicit substitution (ES)** $[x\backslash u]$ on a term t, a **bang** $!t$ to freeze the execution of t, and a **dereliction** $\mathtt{der}(t)$ to fire again the frozen term t. The **argument** of an application $t\,u$ (resp. a closure $t[x\backslash u]$) is the subterm u. From now on, we set $\mathtt{I}_! \coloneqq \lambda z.!z$, $\Delta_! \coloneqq \lambda x.x!x$, and $\Omega_! \coloneqq \Delta_!!\Delta_!$.

Abstractions $\lambda x.t$ and closures $t[x\backslash u]$ bind the variable x in the term t. **Free** and **bound** variables are defined as expected, in particular $\mathtt{fv}(\lambda x.t) \coloneqq \mathtt{fv}(t) \setminus \{x\}$ and $\mathtt{fv}(t[x\backslash u]) \coloneqq \mathtt{fv}(u) \cup (\mathtt{fv}(t) \setminus \{x\})$. The usual notion of α-conversion [20] is extended to $\Lambda_!$, and terms are identified up to α-conversion. We denote by $t\{x\backslash u\}$ the usual (capture avoiding) meta-level substitution of the term u for all free occurrences of the variable x in the term t.

List contexts ($\mathtt{L}$), **surface contexts** ($\mathtt{S}$) and **full contexts** ($\mathtt{F}$), which can be seen as terms containing exactly one **hole** $\diamond$, are inductively defined as follows:

$$\begin{array}{rcl}
\textbf{(List Contexts)} & \mathtt{L} & ::= \ \diamond \mid \mathtt{L}[x\backslash t] \\
\textbf{(Surface Contexts)} & \mathtt{S} & ::= \ \diamond \mid \mathtt{S}\,t \mid t\,\mathtt{S} \mid \lambda x.\mathtt{S} \mid \mathtt{der}(\mathtt{S}) \mid \mathtt{S}[x\backslash t] \mid t[x\backslash \mathtt{S}] \\
\textbf{(Full Contexts)} & \mathtt{F} & ::= \ \diamond \mid \mathtt{F}\,t \mid t\,\mathtt{F} \mid \lambda x.\mathtt{F} \mid \mathtt{der}(\mathtt{F}) \mid \mathtt{F}[x\backslash t] \mid t[x\backslash \mathtt{F}] \mid !\mathtt{F}
\end{array}$$

List and surface contexts are special cases of full contexts. The hole can occur everywhere in full contexts, while it is forbidden under ! in surface contexts. For example, $y\,(\lambda x.\diamond)$ is a surface context hence a full context, while $(!\diamond)[x\backslash \mathtt{I}_!]$ is a full context but not a surface one. We write $\mathtt{F}\langle t\rangle$ for the term obtained by replacing the hole in $\mathtt{F}$ with the term t.

The following **rewrite rules** are the base components of the reduction system of dBang. Any term having the shape of the left-hand side of one of these three rules is called a **redex**.

$$\mathtt{L}\langle \lambda x.t\rangle\, u \mapsto_{\mathtt{dB}} \mathtt{L}\langle t[x\backslash u]\rangle \qquad t[x\backslash \mathtt{L}\langle !u\rangle] \mapsto_{\mathtt{s!}} \mathtt{L}\langle t\{x\backslash u\}\rangle \qquad \mathtt{der}(\mathtt{L}\langle !t\rangle) \mapsto_{\mathtt{d!}} \mathtt{L}\langle t\rangle$$

Rule dB (resp. s!) is assumed to be capture free: no free variable of u (resp. t) is captured by the list context $\mathtt{L}$. The rule dB fires a β-redex and generates an ES. The rule s! operates a substitution provided its argument is a bang: only bang terms can be erased or duplicated, and they lose their bang when the substitution is performed. The rule $\mathtt{d!}$ opens a bang. All these rewrite rules act *at a distance* [7, 9, 2]: the main constructors involved in the rule can be separated by a finite – possibly empty – list context $\mathtt{L}$ of ES. This mechanism unblocks redexes that would otherwise be stuck, *e.g.* $(\lambda x.x)[y\backslash w]!z \mapsto_{\mathtt{dB}} x[x\backslash !z][y\backslash w]$ fires a β-redex where $\mathtt{L} = \diamond[y\backslash w]$ is the list context in between the abstraction $\lambda x.x$ and the argument $!z$.

The **surface reduction** $\to_{\mathrm{S}}$ is the surface closure of the three rewrite rules $\mathtt{dB}$, $\mathtt{s!}$ and $\mathtt{d!}$, *i.e.* $\to_{\mathrm{S}}$ only fires redexes in surface contexts (not under bang). Similarly, the **full reduction** $\to_{\mathrm{F}}$ is the full closure of the three rewrite rules $\mathtt{dB}$, $\mathtt{s!}$ and $\mathtt{d!}$, *i.e.* $\to_{\mathrm{F}}$ fires redexes in any full contexts and thus the bang loses its freezing behavior. For example,

$$(\lambda x.!\mathtt{der}(!x))!y \ \to_{\mathrm{S}} \ (!\mathtt{der}(!x))[x\backslash !y] \ \to_{\mathrm{S}} \ !(\mathtt{der}(!y)) \ \to_{\mathrm{F}} \ !y$$

The first two $\to_{\mathrm{S}}$-steps are $\to_{\mathrm{F}}$-steps too, the last one is not a $\to_{\mathrm{S}}$-step. We denote by $\to_{\mathrm{S}}^*$ the reflexive-transitive closure of $\to_{\mathrm{S}}$, and similarly for $\to_{\mathrm{F}}$. A reduction $\to_{\mathcal{R}}$ is **confluent** if for all t, u_1, u_2 such that $t \to_{\mathcal{R}}^* u_1$ and $t \to_{\mathcal{R}}^* u_2$, there is s such that $u_1 \to_{\mathcal{R}}^* s$ and $u_2 \to_{\mathcal{R}}^* s$.

▶ **Theorem 1.** *The reductions $\to_{\mathrm{S}}$ and $\to_{\mathrm{F}}$ are confluent.*

Proof. For $\to_{\mathrm{S}}$ see [23], for $\to_{\mathrm{F}}$ see [50]. ◀

A term t is a **surface** (resp. **full**) **normal form** if there is no u such that $t \to_{\mathrm{S}} u$ (resp. $t \to_{\mathrm{F}} u$). A term t is **surface** (resp. **full**) **normalizing** if $t \to_{\mathrm{S}}^* u$ (resp. $t \to_{\mathrm{F}}^* u$) for some surface (resp. full) normal form u. Since $\to_{\mathrm{S}} \subsetneq \to_{\mathrm{F}}$, some terms may be surface-normalizing but not full-normalizing, *e.g.* $\lambda x.!(\mathtt{der}(!\Omega_!))$.

As a matter of fact, some ill-formed terms are not redexes but neither represent a desired computation result. They are called **clashes** and have one of the following forms:

$$\mathtt{L}\langle !t\rangle\, u \qquad t[x\backslash \mathtt{L}\langle \lambda x.u\rangle] \qquad \mathtt{der}(\mathtt{L}\langle \lambda x.t\rangle) \qquad t(\mathtt{L}\langle \lambda x.u\rangle) \text{ if } t \neq \mathtt{L}'\langle \lambda y.s\rangle$$

This *static* notion of clash is lifted to a *dynamic* level. A term t is **surface** (resp. **full**) **clash-free** if it does not surface (resp. full) reduce to a term with a clash in surface (resp. full) position, *i.e.* if there are no surface (resp. full) context $\mathtt{S}$ (resp. $\mathtt{F}$) and clash c such that $t \to_{\mathrm{S}}^* \mathtt{S}\langle c\rangle$ (resp. $t \to_{\mathrm{F}}^* \mathtt{F}\langle c\rangle$). For example, $x!(y(\lambda z.z))$ is surface clash-free but not full clash-free as it has a clash $y(\lambda z.z)$ under a bang. Both notions are stable under reduction.

Finally, some terms contain neither redexes nor clashes. A **surface** (resp. **full**) **clash-free normal form** is a surface (resp. full) normal form which is also surface (resp. full) clash-free, as *e.g.* the term xx. These are the results of the computation, and they can even be *syntactically* characterized by the grammar $\mathtt{no}_{\mathrm{S}}$ below.

$$\mathtt{ne}_{\mathrm{S}} \coloneqq x \in \mathcal{X} \mid \mathtt{ne}_{\mathrm{S}}\,\mathtt{na}_{\mathrm{S}} \mid \mathtt{der}(\mathtt{ne}_{\mathrm{S}}) \mid \mathtt{ne}_{\mathrm{S}}[x\backslash \mathtt{ne}_{\mathrm{S}}] \qquad \mathtt{na}_{\mathrm{S}} \coloneqq !t \mid \mathtt{ne}_{\mathrm{S}} \mid \mathtt{na}_{\mathrm{S}}[x\backslash \mathtt{ne}_{\mathrm{S}}]$$
$$\mathtt{nb}_{\mathrm{S}} \coloneqq \mathtt{ne}_{\mathrm{S}} \mid \lambda x.\mathtt{no}_{\mathrm{S}} \mid \mathtt{nb}_{\mathrm{S}}[x\backslash \mathtt{ne}_{\mathrm{S}}] \qquad \mathtt{no}_{\mathrm{S}} \coloneqq \mathtt{na}_{\mathrm{S}} \mid \mathtt{nb}_{\mathrm{S}}$$

▶ **Lemma 2** ([23])**.** *Let $t \in \Lambda_!$, then $t \in \mathtt{no}_{\mathrm{S}}$ iff t is a surface clash-free normal form.*

2.2 Quantitative Typing System

We present the quantitative typing system $\mathcal{B}$ [23], based on [43, 36], for $\mathtt{dBang}$. It contains arrow and intersection types. Intersections are associative, commutative but *not idempotent*, thus an intersection type is represented by a (possibly empty) *finite multiset* $[\sigma_i]_{i\in I}$. Given a countably infinite set $\mathcal{TV}$ of type variables $\alpha, \beta, \gamma, \ldots$, we define by mutual induction:

$$\begin{array}{rrcl} (\textbf{Types}) & \sigma, \tau, \rho & \coloneqq & \alpha \in \mathcal{TV} \mid \mathcal{M} \mid \mathcal{M} \Rightarrow \sigma \\ (\textbf{Multitypes}) & \mathcal{M}, \mathcal{N} & \coloneqq & [\sigma_i]_{i\in I} \text{ where } I \text{ is a finite set} \end{array}$$

$$\frac{}{x : [\sigma] \vdash x : \sigma}\,(var) \qquad \frac{\Gamma \vdash t : \mathcal{M} \Rightarrow \sigma \quad \Delta \vdash u : \mathcal{M}}{\Gamma + \Delta \vdash t\,u : \sigma}\,(app) \qquad \frac{(\Gamma_i \vdash t : \sigma_i)_{i \in I} \quad I \text{ finite}}{+_{i \in I}\Gamma_i \vdash !t : [\sigma_i]_{i \in I}}\,(bg)$$

$$\frac{\Gamma, x : \mathcal{M} \vdash t : \sigma}{\Gamma \vdash \lambda x.t : \mathcal{M} \Rightarrow \sigma}\,(abs) \qquad \frac{\Gamma, x : \mathcal{M} \vdash t : \sigma \quad \Delta \vdash u : \mathcal{M}}{\Gamma + \Delta \vdash t[x\backslash u] : \sigma}\,(es) \qquad \frac{\Gamma \vdash t : [\sigma]}{\Gamma \vdash \mathtt{der}(t) : \sigma}\,(der)$$

Figure 1 Type System $\mathcal{B}$ for the `dBang`-calculus.

A (**type**) **environment**, noted Γ or Δ, is a function from variables to multitypes, assigning the **empty multitype** $[\,]$ to all variables except a finite number (possibly zero). The **empty environment**, noted $\emptyset$, maps every variable to $[\,]$. The **domain** of Γ is $\mathtt{dom}(\Gamma) = \{x \in \mathcal{X} \mid \Gamma(x) \neq [\,]\}$, the **image** of Γ is $\mathtt{im}(\Gamma) = \{\Gamma(x) \mid x \in \mathtt{dom}(\Gamma)\}$. Given the environments Γ and Δ, $\Gamma + \Delta$ is the environment mapping x to $\Gamma(x) \uplus \Delta(x)$, where $\uplus$ denotes multiset union; and $+_{i \in I}\Delta_i$ (with I finite) is its n-ary extension, in particular $+_{i \in I}\Delta_i = \emptyset$ if $I = \emptyset$. An environment Γ is denoted by $x_1 : \mathcal{M}_1, \ldots, x_n : \mathcal{M}_n$ when the x_i's are pairwise distinct variables and $\Gamma(x_i) = \mathcal{M}_i$ for all $1 \leq i \leq n$, and $\Gamma(y) = [\,]$ for $y \notin \{x_1, \ldots, x_n\}$.

A **typing** is a pair $(\Gamma; \sigma)$, where Γ is an environment and σ is a type. A (**typing**) **judgment** is a tuple of the form $\Gamma \vdash t : \sigma$, where $(\Gamma; \sigma)$ is a typing and t is a term (the **subject** of the judgment). The typing system $\mathcal{B}$ for `dBang` is defined by the rules in Figure 1. The axiom rule (var) is relevant, *i.e.* there is no weakening. Rules (abs), (app) and (es) are standard. Rule (bg) has as many premises as elements in the finite (possibly empty) index set I, and its conclusion types $!t$ with a multitype *gathering* all the (possibly different) types in the premises typing t. In particular, when $I = \emptyset$, the rule has no premises, and it types *any* term $!t$ with $[\,]$, leaving the *subterm t untyped*. Rule (der) forces the argument of a dereliction to be typed by a multitype of cardinality 1.

A (**type**) **derivation** in system $\mathcal{B}$ is a tree obtained by applying the rules in Figure 1. The judgment at the root of the type derivation Π is the **conclusion** of Π. We write $\Pi \triangleright_{\mathcal{B}} \Gamma \vdash t : \sigma$ when Π is a derivation in system $\mathcal{B}$ with conclusion $\Gamma \vdash t : \sigma$, and $\triangleright_{\mathcal{B}} \Gamma \vdash t : \sigma$ if there exists some derivation $\Pi \triangleright_{\mathcal{B}} \Gamma \vdash t : \sigma$. A term t is **$\mathcal{B}$-typable** if $\triangleright_{\mathcal{B}} \Gamma \vdash t : \sigma$ for some typing $(\Gamma; \sigma)$.

System $\mathcal{B}$ enjoys subject reduction and expansion with respect to $\rightarrow_{\mathrm{F}}$, and characterizes surface-normalizing clash-free terms.

▶ **Theorem 3** ([23, 13]). *Let $t, u \in \Lambda_!$.*

1. *If $t \rightarrow_{\mathrm{F}} u$, then for any typing $(\Gamma; \sigma)$, one has $\triangleright_{\mathcal{B}} \Gamma \vdash t : \sigma$ if and only if $\triangleright_{\mathcal{B}} \Gamma \vdash u : \sigma$.*
2. *t is $\mathcal{B}$-typable if and only if t surface-reduces to a surface clash-free normal form.*

3 Meaningfulness = Typability + Inhabitation

In this section, we introduce the notion of meaningfulness for `dBang` and we establish a logical characterization of meaningfulness via system $\mathcal{B}$. Intuitively, a term t is meaningful if it can be supplied by some arguments (possibly binding some free variables of t) so that it reduces to some observable term. In `dBang`, the observables are the bang terms since they are the only terms enabling substitution to be fired.

▶ **Definition 4.** *A term t is **`dBang`-meaningful** if there are a testing context $\mathtt{T}$ and $u \in \Lambda_!$ such that $\mathtt{T}\langle t \rangle \rightarrow_{\mathrm{S}}^{*} !u$, where* testing contexts *are defined by the grammar $\mathtt{T} ::= \diamond \mid \mathtt{T}s \mid (\lambda x.\mathtt{T})s$.*[1] *A term t is **`dBang`-meaningless** if it is not `dBang`-meaningful.*

[1] Thanks to a factorization theorem for `dBang` [14], in our definition of `dBang`-meaningfulness $\rightarrow_{\mathrm{S}}^{*}$ can equivalently be replaced by $\rightarrow_{\mathrm{F}}^{*}$. For the same reason, the same remark also applies to Definition 14.

$$\dfrac{\dfrac{}{x : [\mathcal{M} \Rightarrow \sigma] \vdash x : \mathcal{M} \Rightarrow \sigma}\,(var) \qquad \dfrac{}{x : [\mathcal{M}] \vdash x : \mathcal{M}}\,(var)}{x : [\mathcal{M} \Rightarrow \sigma, \mathcal{M}] \vdash xx : \sigma}\,(app)$$

Figure 2 A type derivation of xx in system $\mathcal{B}$.

$$\dfrac{\dfrac{\dfrac{}{x : [\alpha] \vdash x : \alpha}\,(var)}{x : [\alpha] \vdash !x : [\alpha]}\,(bg)}{\emptyset \vdash \lambda x.!x : [\alpha] \Rightarrow [\alpha]}\,(abs)$$

Figure 3 Inhabitation of $[\alpha] \Rightarrow [\alpha]$ in system $\mathcal{B}$.

For example, $\mathtt{I}_!$ is dBang-meaningful, take the testing context $\mathtt{T} = \diamond\, !u$. Both $\Omega_!$ and $x\Omega_!$ are dBang-meaningless: every testing context they are plugged in cannot erase $\Omega_!$, which is not normalizing and does not reduce to a bang term. Note that all testing contexts are surface, and that the hole in a testing context is always in the functional position of an application, in particular if the hole is in the scope of some λ, then this λ must be applied.

Readers familiar with the advanced theory of λ-calculus may wonder about the relevance of our notion of dBang-meaningfulness. In particular, we could have just naively extended the well-known notion of call-by-name *solvability*: a term t is **dBang-solvable** if there are a testing context $\mathtt{T}$ such that $\mathtt{T}\langle t\rangle \rightarrow^*_{\mathtt{S}} \mathtt{I}_!$. We found at least two reasons to not use dBang-solvability: the first one is that we would lose consistency of the smallest $\lambda_{\mathtt{dBang}}$-theory generated by equating all dBang-unsolvable terms (see discussion after Proposition 8), while the second one is that we would lose genericity (see discussion after Corollary 11).

In an adequate calculus, meaningfulness is usually characterized both operationally (normalizability) and logically (typability): a term is meaningful iff it is normalizing for a suitable subreduction of the calculus iff it is typable in a suitable type system. Surprisingly, these characterizations are subtler in dBang, because the language has two (incompatible) data structures: abstractions (playing the role of functions) and bangs (playing as values).

A natural idea to operationally characterize dBang-meaningfulness would be *normalizability* by *surface reduction*, but this fails, even if we require the obtained surface normal form to be clash-free. For instance, the term xx is dBang-meaningless despite being a surface clash-free normal form. Indeed, for xx to be dBang-meaningful, a testing context $\mathtt{T}$ would need to provide a term u to substitute the variable x, so that $\mathtt{T}\langle xx\rangle$ would eventually reduce to a bang. However, achieving this requires the term u to reduce to both an abstraction and a bang, which is impossible. Hence, dBang-meaningfulness is not only the ability to produce a surface clash-free normal form, but also to transform this result into an observable.

Concerning a logical characterization of dBang-meaningfulness, *typability* is not enough, at least in system $\mathcal{B}$, since it just characterizes surface clash-free normalization (Theorem 3.2). For instance, the dBang-meaningless term xx seen above is typable in system $\mathcal{B}$. Every type derivation of xx has the form of that in Figure 2, which reveals the conflict when assigning to x both an arrow type $\mathcal{M} \Rightarrow \sigma$ (the type of terms eventually reducing to abstractions) and a multitype $\mathcal{M}$ (the type of terms eventually reducing to bangs). The inhabitation problem can be used to detect such conflicts, allowing for a handy characterization of meaningfulness. Indeed, the multitype $[\mathcal{M} \Rightarrow \sigma, \mathcal{M}]$ assigned to the variable x in Figure 2 is not *inhabited*. Other (naive and unsuccessful) alternatives are discussed in Section 6.

While it seems complex to syntactically establish operational conditions such as (not) reducing to abstractions or bangs, this is easily achieved semantically. Indeed, we establish a logical characterization of dBang-meaningfulness based on *typability* and *inhabitation* in system $\mathcal{B}$, similarly to what happens in the λ-calculus with pairs [12, 29, 26]. Intuitively, suppose that a term t is dBang-meaningful, so there is a testing context $\mathtt{T}$ such that $\mathtt{T}\langle t\rangle$

reduces to an observable, *i.e.* a bang, which can be (trivially) typed with the typing $(\emptyset; [\,])$ in system $\mathcal{B}$. By Theorem 3.1, $\mathtt{T}\langle t\rangle$ must also be typable by the same typing $(\emptyset; [\,])$, meaning that t is *typable* by some environment $x_1 : \mathcal{M}_1, \ldots, x_n : \mathcal{M}_m$ and some type $\mathcal{N}_1 \Rightarrow \ldots \Rightarrow \mathcal{N}_n \Rightarrow [\,]$, where each of the $\mathcal{M}_i$'s and $\mathcal{N}_i$'s is *inhabited*, *i.e.* there is a term with such a type.

A similar argument holds for other type systems and calculi [29, 26] with their own notions of meaningfulness and observable. The point is to identify the set of types $\mathcal{T}^{\mathtt{obs}}_{\mathcal{S}}$ associated with the observables. In any type system $\mathcal{S}$ whose types are those of Section 2.2, given a set of types $\mathcal{T}^{\mathtt{obs}}_{\mathcal{S}}$ for observable terms, the set of **arguments** $\mathtt{args}_{\mathcal{S}}(\sigma)$ of a type σ is the set of multitypes appearing to the left of arrows, until reaching the type of an observable. Formally, if $\sigma \in \mathcal{T}^{\mathtt{obs}}_{\mathcal{S}}$ then $\mathtt{args}_{\mathcal{S}}(\sigma) := \emptyset$, *otherwise* $\mathtt{args}_{\mathcal{S}}(\alpha) := \emptyset$, $\mathtt{args}_{\mathcal{S}}(\mathcal{M} \Rightarrow \sigma) := \{\mathcal{M}\} \cup \mathtt{args}_{\mathcal{S}}(\sigma)$, and $\mathtt{args}_{\mathcal{S}}(\mathcal{M}) = \emptyset$. In system $\mathcal{B}$, we set $\mathcal{T}^{\mathtt{obs}}_{\mathcal{B}} := \{\mathcal{M} \mid \mathcal{M} \text{ multitype}\}$, because bang terms – the observables in $\mathtt{dBang}$ – can be only typed by multisets. For example, $\mathtt{args}_{\mathcal{B}}([\tau] \Rightarrow (\mathcal{M} \Rightarrow [\alpha])) = \{[\tau], \mathcal{M}\}$. The cases of $\mathtt{dCBN}$ and $\mathtt{dCBV}$ type systems are discussed in Section 5, this is why our definitions deal with a generic type system $\mathcal{S}$.

▶ **Definition 5.** *Let $\mathcal{S}$ be a type system and $\mathtt{inh}_{\mathcal{S}}(\cdot)$ be a predicate on the types of $\mathcal{S}$. A set S of types is **inhabited**, noted $\mathtt{inh}_{\mathcal{S}}(S)$, if $\mathtt{inh}_{\mathcal{S}}(\sigma)$ for all $\sigma \in S$. We write $\mathtt{inh}_{\mathcal{S}}(\Gamma)$ if $\mathtt{inh}_{\mathcal{S}}(\mathtt{im}(\Gamma))$. A typing $(\Gamma; \sigma)$ or a judgment $\Gamma \vdash t : \sigma$ is **$\mathcal{S}$-testable** if $\mathtt{inh}_{\mathcal{S}}(\Gamma)$ and $\mathtt{inh}_{\mathcal{S}}(\mathtt{args}_{\mathcal{S}}(\sigma))$. A term t is **$\mathcal{S}$-testable** if $\triangleright_{\mathcal{S}} \Gamma \vdash t : \sigma$ for some $\mathcal{S}$-testable typing $(\Gamma; \sigma)$.*

A type σ is **inhabited** in system $\mathcal{B}$, noted $\mathtt{inh}_{\mathcal{B}}(\sigma)$, if $\Pi \triangleright_{\mathcal{B}} \emptyset \vdash t : \sigma$ for some Π and t. For instance, in system $\mathcal{B}$, the type $[\,]$ is inhabited by any bang, use rule (bg) with no premises; the environment $\emptyset$ is trivially inhabited; the type $[\alpha] \Rightarrow [\alpha]$ is inhabited, see Figure 3. The term $\lambda x.!x$ is $\mathcal{B}$-testable because $\triangleright_{\mathcal{B}} \emptyset \vdash \lambda x.!x : [\,] \Rightarrow [\,]$ and $(\emptyset, [\,] \Rightarrow [\,])$ is $\mathcal{B}$-testable.

▶ **Lemma 6.** *Let $t \in \Lambda_!$ and $\mathtt{T}$ be a testing context. If $\triangleright_{\mathcal{B}} \emptyset \vdash \mathtt{T}\langle t\rangle : [\,]$, then $\triangleright_{\mathcal{B}} \Gamma \vdash t : \sigma$ with $\mathtt{inh}_{\mathcal{B}}(\Gamma)$ and $\mathtt{inh}_{\mathcal{B}}(\mathtt{args}_{\mathcal{B}}(\sigma))$.*

Inhabitation serves as a crucial tool to produce an observable from a typable term. As said before, any multitype assigned to a variable x by the environment Γ in the derivation of a meaningful term t should be inhabited. Hence, the environment Γ has to be inhabited. However, relying solely on the inhabitation of Γ is not sufficient, as illustrated by the typable term $\triangleright_{\mathcal{B}} \emptyset \vdash \lambda x.xx : [[\mathcal{M}] \Rightarrow \tau, \mathcal{M}] \Rightarrow \tau$, which, despite having a trivially inhabited environment, is $\mathtt{dBang}$-meaningless. We thus also test the inhabitation of type arguments of the type σ of t. This therefore means that $\mathcal{B}$-testability is sufficient to ensure $\mathtt{dBang}$-meaningfulness. Surprisingly, this actually provides a characterization of $\mathtt{dBang}$-meaningfulness.

▶ **Theorem 7** (Logical Characterization)**.** *Let $t \in \Lambda_!$: t is $\mathtt{dBang}$-meaningful iff t is $\mathcal{B}$-testable.*

Now that we have a logical characterization of $\mathtt{dBang}$-meaningfulness, we can reason about the consequences of equating all $\mathtt{dBang}$-meaningless terms in a $\lambda_{\mathtt{dBang}}$-theory, that is, in a quotient of $\Lambda_!$ that roughly equates all terms with the same semantics. Formally, a **$\lambda_{\mathtt{dBang}}$-theory** is an equivalence $\equiv$ on $\Lambda_!$ containing $\to_{\mathrm{F}}$ and closed under full contexts. Let $\mathcal{H}_{\mathtt{dBang}}$ (also noted $\equiv_{\mathcal{H}_{\mathtt{dBang}}}$) be the smallest $\lambda_{\mathtt{dBang}}$-theory equating all $\mathtt{dBang}$-meaningless terms. Theorem 7 entails that $\mathcal{H}_{\mathtt{dBang}}$ is **consistent**, that is, it does not equate all terms.

▶ **Proposition 8** (Consistency of $\mathcal{H}_{\mathtt{dBang}}$)**.** *There exist $t, u \in \Lambda_!$ such that $t \not\equiv_{\mathcal{H}_{\mathtt{dBang}}} u$.*

Replacing $\mathtt{dBang}$-meaningfulness by $\mathtt{dBang}$-solvability would result in the loss of consistency. Indeed, take an arbitrary term $t \in \Lambda_!$ and the two $\mathtt{dBang}$-unsolvable terms $!\Omega_!$ and $\Omega_!$ that the resulting (alternative) theory, written $\mathcal{H}^{\mathtt{solv}}_{\mathtt{dBang}}$, would equate. By contextuality, we would have $(\lambda x.t)\,!\Omega_! \equiv_{\mathcal{H}^{\mathtt{solv}}_{\mathtt{dBang}}} (\lambda x.t)\,\Omega_!$, and by reduction $t \equiv_{\mathcal{H}^{\mathtt{solv}}_{\mathtt{dBang}}} (\lambda x.t)\,!\Omega_!$ (suppose

$x \notin \mathtt{fv}(t))$. Notice that $(\lambda x.t)\,\Omega_!$ is also dBang-unsolvable since the term $\Omega_!$ cannot be erased, thus $(\lambda x.t)\,\Omega_! \equiv_{\mathcal{H}^{\mathtt{solv}}_{\mathtt{dBang}}} \Omega_!$. By transitivity $t \equiv_{\mathcal{H}^{\mathtt{solv}}_{\mathtt{dBang}}} (\lambda x.t)\,!\Omega_! \equiv_{\mathcal{H}^{\mathtt{solv}}_{\mathtt{dBang}}} (\lambda x.t)\,\Omega_! \equiv_{\mathcal{H}^{\mathtt{solv}}_{\mathtt{dBang}}} \Omega_!$. Since t is arbitrary, we easily conclude that all terms are equated in $\mathcal{H}^{\mathtt{solv}}_{\mathtt{dBang}}$, making it inconsistent.

We also corroborate our definition of meaningfulness by proving that it fulfills a pair of genericity properties, and show that $\mathcal{H}_{\mathtt{dBang}}$ admits a unique maximal consistent extension $\mathcal{H}^{*}_{\mathtt{dBang}}$ (Section 4). Finally, we also show that dBang-meaningfulness, $\mathcal{H}_{\mathtt{dBang}}$ and $\mathcal{H}^{*}_{\mathtt{dBang}}$ subsume the well-established corresponding notions for dCBN and dCBV (Section 5).

4 Typed and Surface Genericity in dBang

In Section 3, we proved that dBang-meaningfulness is captured by typability in system $\mathcal{B}$ with some $\mathcal{B}$-testable typing. While this concise characterization formulated as “meaningfulness = typability + inhabitation” [26] provides a high level understanding, its practical manipulation might pose some challenges. Suppose we study some properties of a dBang-meaningful term t through the logical characterization (Theorem 7), thus having a type derivation $\Pi \triangleright_{\mathcal{B}} \Gamma \vdash t : \sigma$ with $(\Gamma;\sigma)$ $\mathcal{B}$-testable. If we proceed by induction on Π, then there is no guarantee that all the judgments appearing in Π have $\mathcal{B}$-testable typings as well, which would make the reasoning awkward and the logical characterization of Theorem 7 difficult to exploit. But this is not the case. Upcoming Lemma 9 states that $\mathcal{B}$-testability propagates bottom-up: if the conclusion of a derivation Π has a $\mathcal{B}$-testable typing, then so does every other judgment in Π.

We write $\Pi \triangleright_{\mathcal{B}_{\mathtt{m}}} \Gamma \vdash t : \sigma$ if $\Pi \triangleright_{\mathcal{B}} \Gamma \vdash t : \sigma$ and each judgment in Π is $\mathcal{B}$-testable, and $\Pi \triangleright_{\mathcal{B}_{\mathtt{m}}} t$ if $\Pi \triangleright_{\mathcal{B}_{\mathtt{m}}} \Gamma \vdash t : \sigma$ holds for some typing $(\Gamma;\sigma)$.

▶ **Lemma 9.** *Let $t \in \Lambda_!$. Then $\Pi \triangleright_{\mathcal{B}} \Gamma \vdash t : \sigma$ with $(\Gamma;\sigma)$ $\mathcal{B}$-testable iff $\Pi \triangleright_{\mathcal{B}_{\mathtt{m}}} \Gamma \vdash t : \sigma$.*

Proof. ($\Leftarrow$): Trivial. ($\Rightarrow$): By an induction on Π. ◀

We can therefore easily use the logical characterization of dBang-meaningfulness to prove the following first *genericity* result for dBang: in a dBang-meaningful term s, a dBang-meaningless subterm can be replaced by any term, without impacting the typing of s.

▶ **Theorem 10** (Typed Genericity). *Let $t \in \Lambda_!$ be dBang-meaningless and $\mathtt{F}$ be a full context. If $\triangleright_{\mathcal{B}_{\mathtt{m}}} \Gamma \vdash \mathtt{F}\langle t\rangle : \sigma$, then $\triangleright_{\mathcal{B}_{\mathtt{m}}} \Gamma \vdash \mathtt{F}\langle u\rangle : \sigma$ for all $u \in \Lambda_!$.*

Proof. By induction on $\mathtt{F}$, using both Theorem 7 and Lemma 9. ◀

This proof relies on the fact that the dBang-meaningless subterm t cannot be explicitly typed in any of the judgments of Π, as typing t in $\mathcal{B}_{\mathtt{m}}$ is equivalent to being dBang-meaningful (by Theorem 7 and Lemma 9). Thus, typed genericity fails when weakening the hypothesis from $\mathcal{B}_{\mathtt{m}}$-typability to $\mathcal{B}$-typability. For example, given the dBang-meaningless term $t = xx$ and the context $\mathtt{F} = y\,\diamond$, $\mathtt{F}\langle t\rangle$ is $\mathcal{B}$-typable as witnessed by $\triangleright_{\mathcal{B}}\, y : [\mathcal{N} \Rightarrow \alpha]\,, x : [\mathcal{M} \Rightarrow \mathcal{N}, \mathcal{M}] \vdash \mathtt{F}\langle t\rangle : \alpha$ – note that the type of x is not inhabited – while $\mathtt{F}\langle \Omega_!\rangle = y\,\Omega_!$ is not $\mathcal{B}$-typable.

As a consequence of typed genericity, we can now prove a qualitative surface genericity result, stating that dBang-meaningless subterms have no bearing on the significance of dBang-meaningful terms: in a dBang-meaningful term s, a dBang-meaningless subterm can be replaced by any term, still keeping s dBang-meaningful. We call this genericity result *surface*, despite it universally quantifies over full contexts, as dBang-meaningful is defined in terms of surface reduction. The corresponding results for dCBN and dCBV are also called *surface* in [15] and *light* in [10], they are both later generalized to a *stratified* notion in [15].

▶ **Corollary 11** (Qualitative Surface Genericity). *Let $\mathtt{F}$ be a full context. If $\mathtt{F}\langle t\rangle$ is dBang-meaningful for some dBang-meaningless $t \in \Lambda_!$, then $\mathtt{F}\langle u\rangle$ is dBang-meaningful for all $u \in \Lambda_!$.*

Proof. Let $u \in \Lambda_!$. As $\mathtt{F}\langle t\rangle$ is $\mathtt{dBang}$-meaningful, then $\Pi \triangleright_{\mathcal{B}_m} \mathtt{F}\langle t\rangle$ holds for some Π by Theorem 7 and Lemma 9. As t is $\mathtt{dBang}$-meaningless, then $\Pi' \triangleright_{\mathcal{B}_m} \mathtt{F}\langle u\rangle$ holds for some Π' by Theorem 10, and hence $\mathtt{F}\langle u\rangle$ is $\mathtt{dBang}$-meaningful by Theorem 7 and Lemma 9. ◀

As for consistency, surface genericity fails when replacing $\mathtt{dBang}$-meaningfulness with $\mathtt{dBang}$-solvability. Indeed, consider the full context $\mathtt{F} := (\lambda y.x)\diamond$ and the two $\mathtt{dBang}$-unsolvable terms $t = !\Omega_!$ and $u = \Omega_!$. One then has that $\mathtt{F}\langle t\rangle = (\lambda y.x)\,!\Omega_! \to_{\mathtt{S}}^* x$ is trivially $\mathtt{dBang}$-solvable, while $\mathtt{F}\langle u\rangle = (\lambda y.x)\,\Omega_!$ is not, as the term $\Omega_!$ cannot be erased.

Genericity is a sanity check on meaningfulness: it holds only if all $\mathtt{dBang}$-meaningless terms are *truly* meaningless. Still, some truly meaningless terms might be misinterpreted as $\mathtt{dBang}$-meaningful. Indeed, when crafting a notion of $\mathtt{dBang}$-meaningless that would satisfy genericity, one might not take *all* truly meaningless terms. The $\lambda_{\mathtt{dBang}}$-theory $\mathcal{H}^*_{\mathtt{dBang}}$ is introduced to avoid that. Let $\mathcal{H}^*_{\mathtt{dBang}}$, also noted $\equiv_{\mathcal{H}^*_{\mathtt{dBang}}}$, be the relation on $\Lambda_!$ defined by:

$$\mathcal{H}^*_{\mathtt{dBang}} := \{(t,u) \mid \forall\ \mathtt{F} \text{ full context, } \mathtt{F}\langle t\rangle\ \mathtt{dBang}\text{-meaningful} \Leftrightarrow \mathtt{F}\langle u\rangle\ \mathtt{dBang}\text{-meaningful}\}$$

The theory $\mathcal{H}^*_{\mathtt{dBang}}$ equates *more* than $\mathcal{H}_{\mathtt{dBang}}$. For example, let $t = x[x\backslash z][y\backslash z]$ and $u = x[y\backslash z][x\backslash z]$: it can be shown that $t \not\equiv_{\mathcal{H}_{\mathtt{dBang}}} u$ while $t \equiv_{\mathcal{H}^*_{\mathtt{dBang}}} u$ due to Theorem 7 since t and u (and so $\mathtt{F}\langle t\rangle$ and $\mathtt{F}\langle u\rangle$ for any full context $\mathtt{F}$) are $\mathcal{B}$-typable by exactly the same typings.

▶ Remark 12. In $\mathcal{H}^*_{\mathtt{dBang}}$, a term reducing to a bang will only be equated to terms which also reduce to bangs. This can be formally proved using a property stating that neutral normal forms can create clashes via a single substitution, technical details can be found in [50].

We expect $\mathcal{H}^*_{\mathtt{dBang}}$ to extend the theory $\mathcal{H}_{\mathtt{dBang}}$. Moreover, to check that all truly meaningless terms are actually $\mathtt{dBang}$-meaningless, we also want this theory to be **maximal**, meaning that no more terms can additionally be equated without compromising consistency.

▶ **Theorem 13.** *$\mathcal{H}^*_{\mathtt{dBang}}$ is the unique maximal consistent $\lambda_{\mathtt{dBang}}$-theory containing $\mathcal{H}_{\mathtt{dBang}}$.*

We now show that the theory $\mathcal{H}^*_{\mathtt{dBang}}$ coincides with the well-known notion of observational equivalence in the literature. Observational equivalence roughly equates terms having the same operational behavior (*i.e.* reduction to an observable) in any context. The fact that $\mathcal{H}^*_{\mathtt{dBang}}$ and observational equivalence coincide means that two different approaches to define a semantics in $\mathtt{dBang}$ actually coincide. This further backs up the idea that what we call $\mathtt{dBang}$-meaningfulness appropriately represents meaningfulness in $\mathtt{dBang}$.

▶ **Definition 14** (Observational Equivalence). *Let $t, u \in \Lambda_!$, then t and u are **open-observational equivalent** (resp. **observational equivalent**), noted $t \cong^o u$ (resp. $t \cong u$) if for every full context $\mathtt{F}$ (resp. full context $\mathtt{F}$ such that $\mathtt{F}\langle t\rangle$ and $\mathtt{F}\langle u\rangle$ are closed), $\mathtt{F}\langle t\rangle \to_{\mathtt{S}}^* !t'$ for some $t' \in \Lambda_!$ iff $\mathtt{F}\langle u\rangle \to_{\mathtt{S}}^* !u'$ for some $u' \in \Lambda_!$.*

Note that, differently from $\cong$, $\cong^o$ quantifies over all full contexts and not only on closing full contexts, hence $\cong^o \subseteq \cong$. Finally, we now prove that the $\lambda_{\mathtt{dBang}}$-theory $\mathcal{H}^*_{\mathtt{dBang}}$ actually coincides with the observational equivalences $\cong$ and $\cong^o$.

▶ **Theorem 15.** *Let $t, u \in \Lambda_!$, then (1) $t \cong u$ iff (2) $t \cong^o u$ iff (3) $t \equiv_{\mathcal{H}^*_{\mathtt{dBang}}} u$.*

Proof. Let $t, u \in \Lambda_!$. Let us show that (3) $\Rightarrow$ (2) $\Rightarrow$ (1) $\Rightarrow$ (3).

- (3) $\Rightarrow$ (2): Let $t \equiv_{\mathcal{H}^*_{\mathtt{dBang}}} u$. Suppose $\mathtt{F}$ is an arbitrary full context such that $\mathtt{F}\langle t\rangle \to_{\mathtt{S}}^* !t'$ for some $t' \in \Lambda_!$. Since $\mathcal{H}^*_{\mathtt{dBang}}$ is a $\lambda_{\mathtt{dBang}}$-theory (Theorem 13) then it is contextual and hence $\mathtt{F}\langle t\rangle \equiv_{\mathcal{H}^*_{\mathtt{dBang}}} \mathtt{F}\langle u\rangle$. By Remark 12, $\mathtt{F}\langle u\rangle \to_{\mathtt{S}}^* !u'$ for some $u' \in \Lambda_!$. Therefore, $t \cong u$.
- (2) $\Rightarrow$ (1): Immediate.
- (1) $\Rightarrow$ (3): We can easily prove that $\cong$ is a consistent $\lambda_{\mathtt{dBang}}$-theory. As (3) $\Rightarrow$ (2) $\Rightarrow$ (1), we have $\cong \supseteq \mathcal{H}^*_{\mathtt{dBang}} \supseteq \mathcal{H}_{\mathtt{dBang}}$ (the last inclusion holds by Theorem 13). By maximality of $\mathcal{H}^*_{\mathtt{dBang}}$ (Theorem 13), then necessarily $\cong \subseteq \mathcal{H}^*_{\mathtt{dBang}}$. ◀

5 Subsuming CBN and CBV Meaningfulness

In this section we show that the notions of meaningfulness for dCBN and dCBV in the literature [15] are subsumed by the one proposed in Section 3 for dBang. We also deduce surface genericity for dCBN and dCBV as a consequence of surface genericity for dBang.

5.1 dCBN and dCBV Calculi

Both dCBN [7, 8, 1] and dCBV [11] are specified using ES and action at a distance, as explained in Section 2.1 for dBang. Both dCBN and dCBV share the same term syntax. The sets Λ of **terms** and Υ of **values** are inductively defined below.

$$\textbf{(Terms)}\quad t,u \;::=\; v \mid t\,u \mid t[x\backslash u] \qquad\qquad \textbf{(Values)}\quad v \;::=\; x \mid \lambda x.t$$

From now on, we set $\mathtt{I} := \lambda z.z$, $\Delta := \lambda x.xx$, and $\Omega := \Delta\Delta$. Note that the syntax contains neither $\mathtt{der}$ nor !. The distinction between terms and values is irrelevant in dCBN but crucial in dCBV. The two calculi also share the same **list contexts** $\mathtt{L_N}, \mathtt{L_V}$ and **full contexts** $\mathtt{F_N}, \mathtt{F_V}$, but use specialized **surface contexts** $\mathtt{S_N}$ and $\mathtt{S_V}$ for dCBN and dCBV, respectively. Again, contexts can be seen as terms with exactly one **hole** $\diamond$ and are inductively defined below.

$$\begin{array}{rrcl}
\textbf{(List Contexts)} & \mathtt{L_N}, \mathtt{L_V} & ::= & \diamond \mid \mathtt{L_N}[x\backslash t] \\
\textbf{(dCBN Surface Contexts)} & \mathtt{S_N} & ::= & \diamond \mid \mathtt{S_N}\,t \mid \lambda x.\mathtt{S_N} \mid \mathtt{S_N}[x\backslash t] \\
\textbf{(dCBV Surface Contexts)} & \mathtt{S_V} & ::= & \diamond \mid \mathtt{S_V}\,t \mid t\,\mathtt{S_V} \mid \mathtt{S_V}[x\backslash t] \mid t[x\backslash \mathtt{S_V}] \\
\textbf{(Full Contexts)} & \mathtt{F_N}, \mathtt{F_V} & ::= & \diamond \mid \mathtt{F_N}\,t \mid t\,\mathtt{F_N} \mid \lambda x.\mathtt{F_N} \mid \mathtt{F_N}[x\backslash t] \mid t[x\backslash \mathtt{F_N}]
\end{array}$$

We now consider the following *rewrite rules*:

$$\mathtt{L_N}\langle \lambda x.t\rangle\, u \mapsto_{\mathtt{dB}} \mathtt{L_N}\langle t[x\backslash u]\rangle \qquad t[x\backslash u] \mapsto_{\mathtt{s}} t\{x\backslash u\} \qquad t[x\backslash \mathtt{L_V}\langle v\rangle] \mapsto_{\mathtt{sV}} \mathtt{L_V}\langle t\{x\backslash v\}\rangle$$

Rules dB and sV are both capture-free: no free variable of u (resp. t) is captured by the list context $\mathtt{L_N}$ (resp. $\mathtt{L_V}$). The differences between dCBN and dCBV are in the previous notions of *surface* contexts, and in the rewrite rules. The **dCBN surface reduction** $\to_{\mathrm{S_N}}$ is the union of the dCBN surface closure of rewrite rules dB and s, while the **dCBV surface reduction** $\to_{\mathrm{S_V}}$ is the union of the dCBV surface closure of the rewrite rules dB and sV. Finally, we use $\to^*_{\mathrm{S_N}}$ (resp. $\to^*_{\mathrm{S_V}}$) to denote the reflexive-transitive closure of the relation $\to_{\mathrm{S_N}}$ (resp. $\to_{\mathrm{S_V}}$).

▶ **Example 16.** For example, $t_0 := (\lambda x.yxx)(\mathtt{II}) \to_{\mathrm{S_N}} (yxx)[x\backslash \mathtt{II}] \to_{\mathrm{S_N}} y(\mathtt{II})(\mathtt{II}) =: t_1$ and $t_0 = (\lambda x.yxx)(\mathtt{II}) \to_{\mathrm{S_V}} (yxx)[x\backslash \mathtt{II}] \to_{\mathrm{S_V}} (yxx)[x\backslash z[z\backslash \mathtt{I}]] \to_{\mathrm{S_V}} (yxx)[x\backslash \mathtt{I}] \to_{\mathrm{S_V}} y\mathtt{II} =: t_2$.

The dCBN surface reduction is (a non-deterministic diamond variant of) the well-known *head* reduction [20], and dCBV surface reduction is the *weak* reduction not reducing under λ's.

The quantitative type systems $\mathcal{N}$ for dCBN and $\mathcal{V}$ for dCBV are presented in Figures 4 and 5, respectively. **Types** and **judgments** are the same as for system $\mathcal{B}$. A derivation Π in system $\mathcal{N}$ with conclusion $\Gamma \vdash t : \sigma$ is noted $\Pi \rhd_{\mathcal{N}} \Gamma \vdash t : \sigma$; we write $\rhd_{\mathcal{N}} \Gamma \vdash t : \sigma$ if there is a derivation $\Pi \rhd_{\mathcal{N}} \Gamma \vdash t : \sigma$. We use similar notations for system $\mathcal{V}$.

The salient property of type systems $\mathcal{N}$ and $\mathcal{V}$ is characterizing normalization in dCBN and dCBV, respectively.

▶ **Lemma 17** ([23, 24])**.** *Let $t \in \Lambda$, then:*
- *t is dCBN surface normalizing iff it is $\mathcal{N}$-typable.*
- *t is dCBV surface normalizing iff it is $\mathcal{V}$-typable.*

$$\frac{}{x:[\sigma]\vdash x:\sigma}\ (var) \qquad \frac{\Gamma\vdash t:[\tau_i]_{i\in I}\Rightarrow\sigma \quad (\Delta_i\vdash u:\tau_i)_{i\in I} \quad I \text{ finite}}{\Gamma+_{i\in I}\Delta_i\vdash t\,u:\sigma}\ (app)$$

$$\frac{\Gamma,x:\mathcal{M}\vdash t:\sigma}{\Gamma\vdash\lambda x.t:\mathcal{M}\Rightarrow\sigma}\ (abs) \qquad \frac{\Gamma,x:[\tau_i]_{i\in I}\vdash t:\sigma \quad (\Delta_i\vdash u:\tau_i)_{i\in I} \quad I \text{ finite}}{\Gamma+_{i\in I}\Delta_i\vdash t[x\backslash u]:\sigma}\ (es)$$

Figure 4 Type System $\mathcal{N}$ for the dCBN-calculus.

$$\frac{}{x:\mathcal{M}\vdash x:\mathcal{M}}\ (var) \qquad \frac{\Gamma\vdash t:[\mathcal{M}\Rightarrow\sigma] \quad \Delta\vdash u:\mathcal{M}}{\Gamma+\Delta\vdash t\,u:\sigma}\ (app)$$

$$\frac{(\Gamma_i,x:\mathcal{M}_i\vdash t:\sigma_i)_{i\in I} \quad I \text{ finite}}{+_{i\in I}\Gamma_i\vdash\lambda x.t:[\mathcal{M}_i\Rightarrow\sigma_i]_{i\in I}}\ (abs) \qquad \frac{\Gamma,x:\mathcal{M}\vdash t:\sigma \quad \Delta\vdash u:\mathcal{M}}{\Gamma+\Delta\vdash t[x\backslash u]:\sigma}\ (es)$$

Figure 5 Type System $\mathcal{V}$ for the dCBV-calculus.

Both dCBN and dCBV can be embedded into dBang by decorating each term with the ! and der modalities. The embedding $\cdot^{\mathtt{n}}$ for dCBN is standard, while various embeddings $\cdot^{\mathtt{v}}$ for dCBV have been proposed in the literature [44, 57, 58, 46, 23, 24, 14], each with its own strengths and weaknesses. In this work, we use the embeddings from [23, 24] defined below:

$$\begin{array}{rcl@{\qquad}rcl}
x^{\mathtt{n}} & := & x & x^{\mathtt{v}} & := & !x \\
(\lambda x.t)^{\mathtt{n}} & := & \lambda x.t^{\mathtt{n}} & (\lambda x.t)^{\mathtt{v}} & := & !\lambda x.t^{\mathtt{v}} \\
(tu)^{\mathtt{n}} & := & t^{\mathtt{n}}\,!u^{\mathtt{n}} & (tu)^{\mathtt{v}} & := & \begin{cases} \mathtt{L}\langle s\rangle\,u^{\mathtt{v}} & \text{if } t^{\mathtt{v}}=\mathtt{L}\langle !s\rangle \\ \mathtt{der}(t^{\mathtt{v}})\,u^{\mathtt{v}} & \text{otherwise}\end{cases} \\
(t[x\backslash u])^{\mathtt{n}} & := & t^{\mathtt{n}}[x\backslash !u^{\mathtt{n}}] & (t[x\backslash u])^{\mathtt{v}} & := & t^{\mathtt{v}}[x\backslash u^{\mathtt{v}}]
\end{array}$$

These translations are extended to contexts as expected by setting $\diamond^{\mathtt{n}} := \diamond$ and $\diamond^{\mathtt{v}} := \diamond$.

▶ **Example 18.** Recalling Example 16, one has $t_0^{\mathtt{n}} = (\lambda x.y\,!x\,!x)\,!(\mathtt{I}_!!\mathtt{I}_!)$, $t_1^{\mathtt{n}} = y\,!(\mathtt{I}_!!\mathtt{I}_!)\,!(\mathtt{I}_!!\mathtt{I}_!)$, $t_0^{\mathtt{v}} = (\lambda x.(\mathtt{der}(y\,!x)\,!x))(\mathtt{I}_!\,!\mathtt{I}_!)$ and $t_2^{\mathtt{v}} = \mathtt{der}(y\,!\mathtt{I}_!)\,!\mathtt{I}_!$.

Let us give some intuition on these embeddings. In dCBN, any argument (right-hand side of application or substitution) can be erased/duplicated, just as bang terms in the dBang-calculus, so that arguments must be translated to bang terms. In dCBV, only values can be erased/duplicated so that values – and only values – must be translated to bang terms. However, this remark alone is not sufficient to achieve a dCBV embedding enjoying good properties, and in particular to translate dCBV-normal forms to dBang-normal forms. The translation of applications is precisely designed in order to guarantee this property.

These embeddings preserve reductions, which will allows us to show that meaningfulness if preserved through embedding (Theorems 25 and 30).

▶ **Lemma 19** (Simulation [23, 24])**.** *Let $t, u \in \Lambda$.*
1. *If $t \to^*_{\mathtt{S}_{\mathtt{N}}} u$ then $t^{\mathtt{n}} \to^*_{\mathtt{S}} u^{\mathtt{n}}$.*
2. *If $t \to^*_{\mathtt{S}_{\mathtt{V}}} u$ then $t^{\mathtt{v}} \to^*_{\mathtt{S}} u^{\mathtt{v}}$.*

▶ **Example 20.** In Example 16, we showed that $t_0 \to^*_{\mathtt{S}_{\mathtt{N}}} t_1$ and $t_0 \to^*_{\mathtt{S}_{\mathtt{V}}} t_2$. Recalling Example 18, one has $t_0^{\mathtt{n}} \to_{\mathtt{S}} (y\,!x\,!x)[x\backslash !(\mathtt{I}_!!\mathtt{I}_!)] \to_{\mathtt{S}} t_1^{\mathtt{n}}$ and $t_0^{\mathtt{v}} \to_{\mathtt{S}} (\mathtt{der}(y\,!x)\,!x)[x\backslash \mathtt{I}_!\,!\mathtt{I}_!] \to_{\mathtt{S}} (\mathtt{der}(y\,!x)\,!x)[x\backslash(!z)[z\backslash !\mathtt{I}_!]] \to_{\mathtt{S}} (\mathtt{der}(y\,!x)\,!x)[x\backslash !\mathtt{I}_!] \to_{\mathtt{S}} t_2^{\mathtt{v}}$.

As the $\mathtt{dCBV}$-embedding uses $\mathtt{der}$, some $\mathtt{d!}$-step might be needed in the simulation process.

These embeddings also preserve typing, which will make possible to project $\mathtt{dBang}$ meaningfulness and surface genericity onto $\mathtt{dCBN}$ and $\mathtt{dCBV}$. More precisely, the two embeddings are proven to be sound and complete with respect to system $\mathcal{B}$.

▶ **Proposition 21** ([23, 24])**.** *Let $t \in \Lambda$ and $(\Gamma; \sigma)$ be a typing.*

1. *One has $\rhd_{\mathcal{N}} \Gamma \vdash t : \sigma$ if and only if $\rhd_{\mathcal{B}} \Gamma \vdash t^{\mathtt{n}} : \sigma$.*
2. *One has $\rhd_{\mathcal{V}} \Gamma \vdash t : \sigma$ if and only if $\rhd_{\mathcal{B}} \Gamma \vdash t^{\mathtt{v}} : \sigma$.*

A straightforward corollary is that $\mathtt{dCBN}$ and $\mathtt{dCBV}$ inhabitation properties are well subsumed in $\mathtt{dBang}$, as illustrated in [13]. In simpler words, any type inhabited in $\mathtt{dCBN}$ (resp. $\mathtt{dCBV}$) is also inhabited in $\mathtt{dBang}$. As expected, the converse is false.

In $\mathtt{dCBV}$ and $\mathtt{dBang}$, typing an arbitrary term and typing an argument is similar, as it can be seen in the right premise $\Delta \vdash u : \mathcal{M}$ of the typing rules (app) and (es) of systems $\mathcal{V}$ and $\mathcal{B}$. This is not the case in $\mathtt{dCBN}$, as the right premise of the (app) and (es) rules of system $\mathcal{N}$ requires, not a *single* derivation, but a finite *set* $(\Delta_i \vdash u : \tau_i)_{i \in I}$ of typing derivation for the same term u. In the logical characterization (Theorem 7), we check that arguments of a given type can be inhabited. We therefore need to reflect the typability of arguments – rather than typability of arbitrary terms – in the definition of $\mathtt{dCBN}$ inhabitation.

▶ **Definition 22.** *In system $\mathcal{N}$, a non-multitype σ is **inhabited**, noted $\mathtt{inh}_{\mathcal{N}}(\sigma)$, if $\Pi \rhd_{\mathcal{N}} \emptyset \vdash t : \sigma$ for some Π and t. A multitype $[\tau_i]_{i \in I}$ is **inhabited** in system $\mathcal{N}$, noted $\mathtt{inh}_{\mathcal{N}}\big([\tau_i]_{i \in I}\big)$ if there exists $u \in \Lambda$ such that for each $i \in I$, $\rhd_{\mathcal{N}} \emptyset \vdash u : \tau_i$.*

*In system $\mathcal{V}$, a type σ is **inhabited**, noted $\mathtt{inh}_{\mathcal{V}}(\sigma)$, if $\Pi \rhd_{\mathcal{V}} \emptyset \vdash t : \sigma$ for some Π and t.*

In particular, the type $[\,]$ is inhabited in both $\mathtt{dCBN}$ and $\mathtt{dCBV}$ (*i.e.* $\mathtt{inh}_{\mathcal{N}}([\,])$ and $\mathtt{inh}_{\mathcal{V}}([\,])$). Similarly, the environment $\emptyset$ is also trivially inhabited in both (*i.e.* $\mathtt{inh}_{\mathcal{N}}(\emptyset)$ and $\mathtt{inh}_{\mathcal{V}}(\emptyset)$).

5.2 $\mathtt{dCBN}$ Meaningfulness and Surface Genericity

In this subsection, our attention shifts towards the $\mathtt{dCBN}$-calculus, where we show that its notion of meaningfulness is subsumed by that of $\mathtt{dBang}$. This observation enables us to project the surface genericity theorem accordingly. We start by introducing $\mathtt{dCBN}$-meaningfulness.

▶ **Definition 23.** *A term $t \in \Lambda$ is **$\mathtt{dCBN}$-meaningful** if there is a testing context $\mathtt{T_N}$ such that $\mathtt{T_N}\langle t \rangle \to^*_{\mathtt{S_N}} \mathtt{I}$, where testing contexts are defined by $\mathtt{T_N} ::= \diamond \mid \mathtt{T_N}\, u \mid (\lambda x.\mathtt{T_N})\, u$.*[2]

For example $t = x(\lambda y.\Omega)$ is $\mathtt{dCBN}$-meaningful as $\mathtt{T_N}\langle t \rangle \to^*_{\mathtt{S_N}} \mathtt{I}$ for $\mathtt{T_N} = (\lambda x.\diamond)(\lambda z.\mathtt{I})$, while Ω and $\lambda x.\Omega$ are $\mathtt{dCBN}$-meaningless as for whatever testing context Ω and $\lambda x.\Omega$ are plugged into, Ω will not be erased. According to the definition of $\mathtt{dCBN}$-meaningfulness, it is natural to define the types of observable terms in $\mathtt{dCBN}$ as the identity types, *i.e.* $\mathcal{T}^{\mathsf{obs}}_{\mathcal{N}} := \{[\sigma] \Rightarrow \sigma \mid \sigma \text{ type}\}$.

Unlike $\mathtt{dBang}$, $\mathtt{dCBN}$-meaningfulness can be characterized both *operationally*, through surface normalizability, and *logically*, through typability in system $\mathcal{N}$. Moreover, this logical characterization turns out to be equivalent to $\mathcal{N}$*-testability*, meaning that $\mathtt{dCBN}$-meaningfulness can also be characterized via typability and inhabitation, as already observed in [27].

---

[2] Usually, $\mathtt{dCBN}$-meaningfulness (aka *solvability*) is defined using contexts of the form $(\lambda x_1 \ldots x_m.\diamond)N_1 \ldots N_n$ $(m, n \geq 0)$ [19, 20, 71], instead of testing contexts. It is easy to check that the two definitions are equivalent in $\mathtt{dCBN}$. The benefit of our definition is that the same testing contexts are also used to define $\mathtt{dCBV}$-meaningfulness (Section 5.3).

▶ **Theorem 24** (Characterizations of dCBN-Meaningfulness [29, 27, 23]). *Let $t \in \Lambda$.*

1. ***(Operational)*** *t is dCBN-meaningful iff t is dCBN surface-normalizing.*
2. ***(Logical)*** *(1) t is dCBN-meaningful iff (2) t is $\mathcal{N}$-typable iff (3) t is $\mathcal{N}$-testable.*

Thanks to the specific shape of dCBN-normal forms, we can always type a dCBN-meaningful term t by a typing $(\Gamma; \sigma)$ such that the *non-empty* multitypes in Γ and $\mathtt{args}_{\mathcal{N}}(\sigma)$ are of the form $[[\,] \Rightarrow \cdots [\,] \Rightarrow [\alpha] \Rightarrow \alpha]$. These types are trivially inhabited by erasers of the form $\lambda x_1. \cdots \lambda x_n.\mathtt{I}$, used to prove that $\mathcal{N}$-typability implies dCBN-meaningfulness.

Having an operational characterization of meaningfulness seems to point out that transforming a result into something observable is a trivial operation in dCBN. Indeed, using simulation (Lemma 19.2), we easily show that dCBN-meaningful is preserved by the dCBN-embedding, thus confirming this intuition. Moreover, and thanks to the logical characterization (Theorem 24.2), we show that the converse also holds, yielding the following result.

▶ **Theorem 25.** *Let $t \in \Lambda$, then t is dCBN-meaningful iff $t^{\mathtt{n}}$ is dBang-meaningful.*

Proof.

($\Rightarrow$) We present here an operational proof. Let t be dCBN-meaningful, thus $\mathtt{T_N}\langle t \rangle \to^*_{\mathtt{S_N}} \mathtt{I}$ for some testing context $\mathtt{T_N}$. By induction on $\mathtt{T_N}$, one has that $(\mathtt{T_N}\langle t \rangle)^{\mathtt{n}} = \mathtt{T_N^n}\langle t^{\mathtt{n}} \rangle$. By simulation (Lemma 19.1), one deduces that $\mathtt{T_N^n}\langle t^{\mathtt{n}} \rangle \to^*_{\mathtt{S}} \lambda x.x$ thus $\mathtt{T_N^n}\langle t^{\mathtt{n}} \rangle\, !!y \to^*_{\mathtt{S}} (\lambda x.x)!!y \to^*_{\mathtt{S}} !y$. Notice that $\mathtt{T_N^n}!!y$ is a dBang-testing context. We thus conclude that $t^{\mathtt{n}}$ is dBang-meaningful.

($\Leftarrow$) Let $t^{\mathtt{n}}$ be dBang-meaningful, then using Theorem 7, it is $\mathcal{B}$-testable and thus $\mathcal{B}$-typable. By Proposition 21.1, t is $\mathcal{N}$-typable and hence t is dCBN-meaningful by Theorem 24. ◀

Observe for example that $\mathtt{I}$ and $\mathtt{I}^{\mathtt{n}} = \mathtt{I}_!$ are both dCBN/dBang-meaningful while Ω and $\Omega^{\mathtt{n}} = \Omega_!$ are both dCBN/dBang-meaningless.

Theorem 25 states that dCBN-meaningfulness precisely aligns with dBang-meaningfulness on its image via $\cdot^{\mathtt{n}}$, strengthening the idea that these two notions are adequately chosen. Thanks to Theorem 25, we can now project surface genericity from dBang to dCBN.

▶ **Theorem 26** (dCBN Qualitative Surface Genericity). *Let $\mathtt{F_N}$ be a full context. If $\mathtt{F_N}\langle t \rangle$ is dCBN-meaningful for some dCBN-meaningless $t \in \Lambda$, then $\mathtt{F_N}\langle u \rangle$ is dCBN-meaningful for every $u \in \Lambda$.*

Proof. Let $t \in \Lambda$ be dCBN-meaningless and $\mathtt{F_N}$ be a full context. Suppose that $\mathtt{F_N}\langle t \rangle$ is dCBN-meaningful: by Theorem 25 and since $(\mathtt{F_N}\langle t \rangle)^{\mathtt{n}} = \mathtt{F_N^n}\langle t^{\mathtt{n}} \rangle$ (simple induction on $\mathtt{F_N}$), $\mathtt{F_N^n}\langle t^{\mathtt{n}} \rangle$ is dBang-meaningful, and $t^{\mathtt{n}}$ is dBang-meaningless. By Corollary 11, for any $u \in \Lambda$, $\mathtt{F_N^n}\langle u^{\mathtt{n}} \rangle = (\mathtt{F_N}\langle u \rangle)^{\mathtt{n}}$ is dBang-meaningful, and hence $\mathtt{F_N}\langle u \rangle$ is dCBN-meaningful using Theorem 25. ◀

We now discuss some crucial consequences of our previous results, captured by the use of $\lambda_{\mathtt{dCBN}}$-theories. A $\lambda_{\mathtt{dCBN}}$**-theory** is an equivalence $\equiv$ on Λ containing $\to_{\mathtt{F_N}}$ and closed under full contexts. Let $\mathcal{H}_{\mathtt{dCBN}}$ (also noted $\equiv_{\mathcal{H}_{\mathtt{dCBN}}}$) be the smallest $\lambda_{\mathtt{dCBN}}$-theory equating all dCBN-meaningless terms, and let $\mathcal{H}^*_{\mathtt{dCBN}}$ be defined as follows:

$$\mathcal{H}^*_{\mathtt{dCBN}} \coloneqq \{(t, u) \mid \forall\, \mathtt{F_N} \text{ full context, } \mathtt{F_N}\langle t \rangle \text{ dCBN-meaningful} \Leftrightarrow \mathtt{F}\langle u \rangle \text{ dBang-meaningful}\}$$

As for dBang, $\mathcal{H}^*_{\mathtt{dCBN}}$ is the maximal consistent $\lambda_{\mathtt{dCBN}}$-theory containing $\mathcal{H}_{\mathtt{dCBN}}$ and it coincides with observational equivalence in dCBN (see [15]). Thanks to the preservation of meaningfulness via the dCBN-embedding $\cdot^{\mathtt{n}}$ (Theorem 25), we can actually relate the theories $\mathcal{H}_{\mathtt{dBang}}$ and $\mathcal{H}^*_{\mathtt{dBang}}$ (in dBang) to the corresponding ones in dCBN, that is, $\mathcal{H}_{\mathtt{dCBN}}$ and $\mathcal{H}^*_{\mathtt{dCBN}}$ respectively.

▶ **Theorem 27.** *Let $t, u \in \Lambda$.*

1. *If $t \equiv_{\mathcal{H}_{\mathtt{dCBN}}} u$ then $t^{\mathtt{n}} \equiv_{\mathcal{H}_{\mathtt{dBang}}} u^{\mathtt{n}}$.*
2. *If $t^{\mathtt{n}} \equiv_{\mathcal{H}^*_{\mathtt{dBang}}} u^{\mathtt{n}}$ then $t \equiv_{\mathcal{H}^*_{\mathtt{dCBN}}} u$.*

Proof.

1. Immediate consequence of Theorem 25 and Lemma 19.1.
2. Let $t, u \in \Lambda$ such that $t^{\mathtt{n}} \equiv_{\mathcal{H}^*_{\mathtt{dBang}}} u^{\mathtt{n}}$. Let $\mathtt{F_N}$ be a full context and suppose that $\mathtt{F_N}\langle t\rangle$ is dCBN-meaningful. Using Theorem 25, one deduces that $(\mathtt{F_N}\langle t\rangle)^{\mathtt{n}} = \mathtt{F}^{\mathtt{n}}_{\mathtt{N}}\langle t^{\mathtt{n}}\rangle$ is dBang-meaningful. Since $t^{\mathtt{n}} \equiv_{\mathcal{H}^*_{\mathtt{dBang}}} u^{\mathtt{n}}$, one has that $\mathtt{F}^{\mathtt{n}}_{\mathtt{N}}\langle u^{\mathtt{n}}\rangle = (\mathtt{F_N}\langle u\rangle)^{\mathtt{n}}$ is dBang-meaningful. Using Theorem 25, one concludes that $\mathtt{F_N}\langle u\rangle$ is dCBN-meaningful and therefore $t \equiv_{\mathcal{H}^*_{\mathtt{dCBN}}} u$. ◀

We strongly conjecture that the converse of Theorem 27.1 also holds. Perhaps unexpectedly, the converse of Theorem 27.2 is actually false. Indeed, η-expansion is included in $\mathcal{H}^*_{\mathtt{dCBN}}$ (see [20]) but not in $\mathcal{H}^*_{\mathtt{dBang}}$ thus $x \equiv_{\mathcal{H}^*_{\mathtt{dCBN}}} \lambda y.xy$ but $x^{\mathtt{n}} = x \not\equiv_{\mathcal{H}^*_{\mathtt{dCBN}}} \lambda y.x!y = (\lambda y.xy)^{\mathtt{n}}$: the context $\mathtt{F} = \diamond[x\backslash !!w]$ separates x and $\lambda y.x!y$. However, through Theorem 15, this phenomenon is not so surprising as it tells us that the dCBN observational equivalence does not coincide with dBang observational equivalence on the image of $\cdot^{\mathtt{n}}$, since dBang is a finer language than dCBN, with more contexts to separate terms operationally.

5.3 dCBV Meaningfulness and Surface Genericity

We now move to the dCBV-calculus, where we show that its notion of meaningfulness is subsumed by that of the dBang-calculus, and then project surface genericity theorem accordingly.

Adapting meaningfulness from dCBN to dCBV by replacing dCBN-reduction with dCBV-reduction may seem initially promising. This notion, known as dCBV-solvability, has appealing properties [64, 71, 11, 30, 47, 6]. Unfortunately, Accattoli and Guerrieri showed that genericity fails in such setting [6], and that equating unsolvable terms yields an inconsistent theory (see *e.g.* [6]). Consequently, dCBV-meaningfulness cannot be identified with dCBV-solvability. Identifying appropriate notions to capture dCBV meaningful λ-terms and formally validating these notions has been a longstanding and challenging open question.

Paolini and Ronchi Della Rocca [64, 71] introduced the notion of *potentially valuability* for CBV, also studied in [63, 11, 30, 42] and renamed *(dCBV) scrutability* in [6]. This notion, which we introduce below, proves to be suitable dCBV-meaningfulness. Notably, it aligns seamlessly with dBang-meaningfulness through the dCBV-embedding and thus enjoys a genericity theorem.

▶ **Definition 28.** *A term $t \in \Lambda$ is **dCBV-meaningful** if there exists a testing context $\mathtt{T_V}$ and a value v such that $\mathtt{T_V}\langle t\rangle \to^*_{\mathtt{S_V}} v$, where testing contexts are defined by $\mathtt{T_V} ::= \diamond \mid \mathtt{T_V}\, u \mid (\lambda x.\mathtt{T_V})\, u$.*

For example $t = x(\lambda y.z)$ is dCBV-meaningful as $\mathtt{T_V}\langle t\rangle \to^*_{\mathtt{S_V}} \lambda y.z$ for $\mathtt{T_V} = (\lambda x.\diamond)(\lambda z.z)$, while Ω and $x\Omega$ are dCBV-meaningless as for whatever testing context Ω and $x\Omega$ are plugged into, Ω will not be erased. Note that the set of testing contexts is the same as those of dCBN.

Notice that this definition closely mirrors that of dBang-meaningfulness, with the primary difference being the replacement of dBang values for those of dCBV. Since values are typed with multitypes, it is natural to take them as types of the observable terms in dCBV (*i.e.* $\mathcal{T}^{\mathsf{obs}}_{\mathcal{V}} := \{\mathcal{M} \mid \mathcal{M} \text{ multitype}\}$). Consequently, and thanks to the preservation of typing (Proposition 21.2), one easily shows that testability is preserved through the dCBV translation: if a term t is $\mathcal{V}$-testable, then its image $t^{\mathtt{v}}$ is $\mathcal{B}$-testable.

As in dCBN and unlike dBang, dCBV-meaningfulness can actually be characterized both *operationally*, through surface normalizability, and *logically*, through typability in system $\mathcal{V}$. Moreover, the logical characterization turns out to be equivalent to $\mathcal{V}$-testability, meaning that dCBV-meaningfulness is also characterized by means of typability and inhabitation.

▶ **Theorem 29** (Characterizations of dCBV-Meaningfulness [11, 6, 23])**.** *Let $t \in \Lambda$.*

1. ***(Operational)** t is dCBV-meaningful iff t is dCBV surface-normalizing.*
2. ***(Logical)** (1) t is dCBV-meaningful iff (2) t is $\mathcal{V}$-typable iff (3) t is $\mathcal{V}$-testable.*

The notion of observable aligns in $\mathtt{dCBV}$ and $\mathtt{dBang}$, at least from the type perspective. This yields a simple fully semantical proof of the preservation of $\mathtt{dCBV}$-meaningfulness.

▶ **Theorem 30.** *Let $t \in \Lambda$, then t is $\mathtt{dCBV}$-meaningful iff $t^{\mathtt{v}}$ is $\mathtt{dBang}$-meaningful.*

Proof.

($\Rightarrow$) We present here a semantical proof. Let t be $\mathtt{dCBV}$-meaningful, then using Theorem 29, one has that t is $\mathcal{V}$-testable thus, by preservation of testability, $t^{\mathtt{v}}$ is $\mathcal{B}$-testable and one concludes that $t^{\mathtt{v}}$ is $\mathtt{dBang}$-meaningful according to Theorem 7.

($\Leftarrow$) Let $t^{\mathtt{v}}$ be $\mathtt{dBang}$-meaningful, then using Theorem 7, it is $\mathcal{B}$-testable thus $\mathcal{B}$-typable. By Proposition 21.2, t is $\mathcal{V}$-typable and thus t is $\mathtt{dCBV}$-meaningful by Theorem 29. ◀

Observe for example that $\mathtt{I}$ and $\mathtt{I}^{\mathtt{v}} = !\mathtt{I}_!$ are both $\mathtt{dCBV}/\mathtt{dBang}$-meaningful while Ω and $\Omega^{\mathtt{v}} = \Omega_!$ are both $\mathtt{dCBV}/\mathtt{dBang}$-meaningless.

Theorem 30 states that $\mathtt{dCBV}$-meaningfulness precisely aligns with $\mathtt{dBang}$-meaningfulness on its image, strengthening the idea that these two notions are adequately chosen. Thanks to Theorem 30, we can now project surface genericity from $\mathtt{dBang}$ to $\mathtt{dCBV}$.

▶ **Theorem 31** (dCBV Qualitative Surface Genericity)**.** *Let $\mathtt{F}_{\mathtt{V}}$ be a full context. If $\mathtt{F}_{\mathtt{V}}\langle t\rangle$ is $\mathtt{dCBV}$-meaningful for some $\mathtt{dCBV}$-meaningless $t \in \Lambda$, then $\mathtt{F}_{\mathtt{V}}\langle u\rangle$ is $\mathtt{dCBV}$-meaningful for every $u \in \Lambda$.*

Proof. Let $t \in \Lambda$ be $\mathtt{dCBV}$-meaningless and $\mathtt{F}_{\mathtt{V}}$ be a full context. Suppose that $\mathtt{F}_{\mathtt{V}}\langle t\rangle$ is $\mathtt{dCBV}$-meaningful, then using Theorem 30, $(\mathtt{F}_{\mathtt{V}}\langle t\rangle)^{\mathtt{v}}$ is $\mathtt{dBang}$-meaningful, and $t^{\mathtt{v}}$ is $\mathtt{dBang}$-meaningless. By induction on $\mathtt{F}_{\mathtt{V}}$, $(\mathtt{F}_{\mathtt{V}}\langle t\rangle)^{\mathtt{v}} = \mathtt{F}_{\mathtt{V}}^{\mathtt{v}}\langle t^{\mathtt{v}}\rangle$ thus $\mathtt{F}_{\mathtt{V}}^{\mathtt{v}}\langle t^{\mathtt{v}}\rangle$ is $\mathtt{dBang}$-meaningful. By Corollary 11, for any $u \in \Lambda$, $\mathtt{F}_{\mathtt{V}}^{\mathtt{v}}\langle u^{\mathtt{v}}\rangle$ is $\mathtt{dBang}$-meaningful. So, by typing preservation (Proposition 21.2), $(\mathtt{F}_{\mathtt{V}}\langle u\rangle)^{\mathtt{v}}$ is $\mathtt{dBang}$-meaningful, and hence $\mathtt{F}_{\mathtt{V}}\langle u\rangle$ is $\mathtt{dCBV}$-meaningful using Theorem 30. ◀

We now discuss some crucial consequences of our previous results, captured by the use of $\lambda_{\mathtt{dCBV}}$-theories. A **$\lambda_{\mathtt{dCBV}}$-theory** is an equivalence $\equiv$ on Λ containing $\rightarrow_{\mathtt{F}_{\mathtt{V}}}$ and closed under full contexts. Let $\mathcal{H}_{\mathtt{dCBV}}$ (also noted $\equiv_{\mathcal{H}_{\mathtt{dCBV}}}$) be the smallest $\lambda_{\mathtt{dCBV}}$-theory equating all $\mathtt{dCBV}$-meaningless terms, and let $\mathcal{H}^*_{\mathtt{dCBV}}$ be defined as follows:

$$\mathcal{H}^*_{\mathtt{dCBV}} \coloneqq \{(t,u) \mid \forall\, \mathtt{F}_{\mathtt{V}} \text{ full context, } \mathtt{F}_{\mathtt{V}}\langle t\rangle\ \mathtt{dCBV}\text{-meaningful} \Leftrightarrow \mathtt{F}\langle u\rangle\ \mathtt{dBang}\text{-meaningful}\}$$

As for $\mathtt{dBang}$ and $\mathtt{dCBN}$, $\mathcal{H}^*_{\mathtt{dCBV}}$ is the maximal consistent $\lambda_{\mathtt{dCBV}}$-theory containing $\mathcal{H}_{\mathtt{dCBV}}$ and coincides with observational equivalence in $\mathtt{dCBV}$ (see [15]). Again, thanks to the preservation of meaningfulness via the $\mathtt{dCBV}$-embedding $\cdot^{\mathtt{v}}$ (Theorem 30), we can relate the theories $\mathcal{H}_{\mathtt{dBang}}$ and $\mathcal{H}^*_{\mathtt{dBang}}$ (in $\mathtt{dBang}$) to the corresponding ones in $\mathtt{dCBV}$, that is, $\mathcal{H}_{\mathtt{dCBN}}$ and $\mathcal{H}^*_{\mathtt{dCBV}}$.

▶ **Theorem 32.** *Let $t, u \in \Lambda$.*

1. *If $t \equiv_{\mathcal{H}_{\mathtt{dCBV}}} u$ then $t^{\mathtt{v}} \equiv_{\mathcal{H}_{\mathtt{dBang}}} u^{\mathtt{v}}$.*
2. *If $t^{\mathtt{v}} \equiv_{\mathcal{H}^*_{\mathtt{dBang}}} u^{\mathtt{v}}$ then $t \equiv_{\mathcal{H}^*_{\mathtt{dCBV}}} u$.*

Proof.

1. Immediate consequence of Theorem 30 and Lemma 19.2
2. Let $t, u \in \Lambda$ such that $t^{\mathtt{v}} \equiv_{\mathcal{H}^*_{\mathtt{dBang}}} u^{\mathtt{v}}$. Let $\mathtt{F}_{\mathtt{V}}$ be a full context and suppose that $\mathtt{F}_{\mathtt{V}}\langle t\rangle$ is $\mathtt{dCBV}$-meaningful. By Theorem 30, $(\mathtt{F}_{\mathtt{V}}\langle t\rangle)^{\mathtt{v}}$ is $\mathtt{dBang}$-meaningful, and by Theorem 7 $(\mathtt{F}_{\mathtt{V}}\langle t\rangle)^{\mathtt{v}}$ is $\mathcal{B}$-testable. As $(\mathtt{F}_{\mathtt{V}}\langle t\rangle)^{\mathtt{v}} \rightarrow^*_{\mathtt{F}} \mathtt{F}_{\mathtt{V}}^{\mathtt{v}}\langle t^{\mathtt{v}}\rangle$ (see [14]) and typing is preserved by reduction (Theorem 3.1), we deduce that $\mathtt{F}_{\mathtt{V}}^{\mathtt{v}}\langle t^{\mathtt{v}}\rangle$ is also $\mathtt{dBang}$-meaningful. Since $t^{\mathtt{v}} \equiv_{\mathcal{H}^*_{\mathtt{dBang}}} u^{\mathtt{v}}$, one has that $\mathtt{F}_{\mathtt{V}}^{\mathtt{v}}\langle u^{\mathtt{v}}\rangle$ is $\mathtt{dBang}$-meaningful and so is $(\mathtt{F}_{\mathtt{V}}\langle u\rangle)^{\mathtt{v}}$, thanks to Theorem 3.1 and since $(\mathtt{F}_{\mathtt{V}}\langle u\rangle)^{\mathtt{v}} \rightarrow^*_{\mathtt{F}} \mathtt{F}_{\mathtt{V}}^{\mathtt{v}}\langle u^{\mathtt{v}}\rangle$. By Theorem 30, $\mathtt{F}_{\mathtt{V}}\langle u\rangle$ is $\mathtt{dCBV}$-meaningful and hence $t \equiv_{\mathcal{H}^*_{\mathtt{dCBV}}} u$. ◀

As in the dCBN-case, we strongly conjecture that the converse of Theorem 32.1 also holds. As in the dCBN-case again, the converse of Theorem 32.2 is actually false. Indeed, the η_v-expansion is included in $\mathcal{H}^*_{\mathtt{dCBV}}$ (see [66, 53, 22, 3]) but not in $\mathcal{H}^*_{\mathtt{dBang}}$, *i.e.* $x \equiv_{\mathcal{H}^*_{\mathtt{dCBV}}} \lambda y.xy$ but $x^{\mathtt{v}} = !x \not\equiv_{\mathcal{H}^*_{\mathtt{dCBV}}} !\lambda y.x!y = (\lambda y.xy)^{\mathtt{v}}$: the context $\mathtt{F} = \mathtt{der}(\diamond)\,[x\backslash !!w]$ separates the two. Again, from the viewpoint of Theorem 15, this phenomenon is not so surprising as it tells us that the dCBV and dBang observational equivalence does not coincide on the image of $\cdot^{\mathtt{v}}$, since dBang is a finer language than dCBV, with more contexts to separate terms operationally.

6 Conclusion and Future Work

We defined a notion of meaningful term, in a unifying well-established framework dBang that is able to capture both dCBN and dCBV calculi. We validated this notion of meaningfulness by providing a (high-level) characterization based on both typability and inhabitation, and showing a (surface) genericity result. All these results in dBang are perfectly analogous to well-known results for dCBN and dCBV [15]. Furthermore, both meaningfulness and genericity in dBang are shown to capture their respective notions in dCBN and dCBV. This suggests that there is a sort of *canonicity* in our definition of dBang-meaningfulness.

It is natural to wonder why this work is not conducted on the usual CBN and CBV calculi but rather their distant version dCBN and dCBV, which make use of explicit substitutions. The main reason is the *non-adequacy* of Plotkin's CBV calculus [66], meaning that some observational equivalent terms have different operational behaviors. Indeed, take the term $t := (\lambda x.\Delta)\,(yy)\,\Delta$ which is observationally equivalent to the prototypical diverging term Ω. Since $\lambda x.\Delta$ is applied to yy – which is not a value and cannot reduce to a value – it makes t a normal form in Plotkin's CBV. This mismatch complicates the study of dCBV-meaningfulness. Notice that this issue is solved in dCBV as the term t now diverges: $t \to_{\mathsf{S_v}} \Delta[x\backslash yy]\Delta \to_{\mathsf{S_v}} (zz)[z\backslash\Delta][x\backslash yy] \to_{\mathsf{S_v}} \Omega[x\backslash yy] \to_{\mathsf{S_v}} \ldots$, as expected. Furthermore, the observational equivalences generated by Plotkin's CBV and dCBV coincide, making the calculus switch harmless. Since adequacy for CBV is recovered thanks to ES and action at a distance, it is then natural to adopt a similar specification for CBN, knowing that standard CBN λ-calculus and dCBN are operationally and semantically equivalent.

While the logical characterization of meaningfulness for dBang (Theorem 7) requires additional hypotheses (typability and inhabitation) compared to those for dCBN (Theorem 24) and dCBV (Theorem 29), which only require typability, this dissimilarity should not be mistakenly interpreted as a weakness of our approach.

Firstly, the inhabitation condition becomes trivial in the case of dCBN and dCBV, as testability and typability coincide in both cases. Consequently, our approach to meaningfulness for dBang clearly provides a conservative extension of those for dCBN and dCBV.

Secondly, the use of distinct term constructors to specify data that cannot be intermingled seems unavoidable to embed both call-by-name and call-by-value calling paradigms within a single unifying framework. In the case of dBang, a clear distinction must be made between functions (represented by abstractions) and duplicable terms (represented by bang). This syntactic distinction, absent in both call-by-name and call-by-value, results in a unifying framework containing (at least) two built-in primitives that capture *incompatible* data. Then, the use of intersection types enable, in principle, such mismatch to exists even though a term cannot actually be a bang and a function at the same time. To address this issue, it may seem tempting to explore some syntactical restriction of intersection type systems such as uniformity [65] or compatibility [29], but both these cases result in a loss of completeness.

Finally, the characterization of meaningfulness through typability and inhabitation in a language equipped with incompatible data structures was initially studied in [29, 26], in the context of a λ-calculus with pair patterns. Clearly, functions cannot be pattern-matched by pair patterns, and pairs cannot be applied to arguments.

Besides that, several questions remain to be explored. First of all, we aim to show that our notion of meaningfulness for `dBang` allows us to prove a *full* genericity result in `dBang` in Barendregt's sense as mentioned in Section 1 (meaningless subterms are computationally irrelevant in the evaluation of full normalizing terms). A notion of stratified reduction, a finer operational semantics generalizing surface reduction to different levels, has been recently defined for `dCBN` and `dCBV` [15]. Stratified reduction is a key tool to show a full genericity result for both `dCBN` and `dCBV`. We plan to transfer these techniques to the more general framework of `dBang`, so that full genericity for `dCBN` and `dCBV` can be simply obtained by projecting the more general notion of full genericity for `dBang` via `CBN`/`CBV` translations.

It has been observed [69] that `dBang` can be embedded in this pattern language. Nevertheless, these two languages are not semantically equivalent, as `dBang` allows only duplication of values (bang terms), whereas the pattern language allows duplication of arbitrary terms.

We also plan to further study the properties of the smallest theory $\mathcal{H}_{\texttt{dBang}}$ generated by equating all the meaningful terms in `dBang`. We strongly conjecture that $\mathcal{H}_{\texttt{dBang}}$ restricted to the image of the embedding $\cdot^{\texttt{n}}$ (resp. $\cdot^{\texttt{v}}$) is equivalent to $\mathcal{H}_{\texttt{dCBN}}$ in `dCBN` (resp. $\mathcal{H}_{\texttt{dCBV}}$ in `dCBV`).

We would like to extend our study to other natural objects in the theory of programming, such as Böhm trees for `dBang` and their related theorems (*e.g.* approximation and separability). Böhm trees for `dBang` are expected to encompass both `dCBN` [20] and `dCBV` [49] ones.

Unifying frameworks such as `dBang` should also provide other general results for `dCBN` and `dCBV`, such as standardization, separability, etc. All this is left to future work. Finally, a more ambitious goal would be to generalize these results to models of computations with effects, such as global memory, non-determinism, exceptions, etc. This would approach our study on `dBang` to a more general unifying framework such as call-by-push-value [54, 55].

References

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

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

3 Beniamino Accattoli, Claudia Faggian, and Adrienne Lancelot. Normal form bisimulations by value. *CoRR*, abs/2303.08161, 2023. `doi:10.48550/arXiv.2303.08161`.

4 Beniamino Accattoli, Stéphane Graham-Lengrand, and Delia Kesner. Tight typings and split bounds, fully developed. *Journal of Functional Programming*, 30:e14, 2020. `doi:10.1017/S095679682000012X`.

5 Beniamino Accattoli and Giulio Guerrieri. Call-by-value solvability and multi types. *CoRR*, abs/2202.03079, 2022. `arXiv:2202.03079`.

6 Beniamino Accattoli and Giulio Guerrieri. The theory of call-by-value solvability. *Proceedings of the ACM on Programming Languages*, 6(ICFP):855–885, 2022. `doi:10.1145/3547652`.

7 Beniamino Accattoli and Delia Kesner. The structural *lambda*-calculus. In *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 381–395. Springer, 2010. `doi:10.1007/978-3-642-15205-4_30`.

8 Beniamino Accattoli and Delia Kesner. The permutative λ-calculus. In Nikolaj S. Bjørner and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 18th International Conference, LPAR-18, Mérida, Venezuela, March 11-15, 2012. Proceedings*, volume 7180 of *Lecture Notes in Computer Science*, pages 23–36. Springer, 2012. doi:10.1007/978-3-642-28717-6_5.

9 Beniamino Accattoli and Delia Kesner. Preservation of strong normalisation modulo permutations for the structural lambda-calculus. *Log. Methods Comput. Sci.*, 8(1), 2012. doi:10.2168/LMCS-8(1:28)2012.

10 Beniamino Accattoli and Adrienne Lancelot. Light genericity. In Naoki Kobayashi and James Worrell, editors, *Foundations of Software Science and Computation Structures - 27th International Conference, FoSSaCS 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6-11, 2024, Proceedings, Part II*, volume 14575 of *Lecture Notes in Computer Science*, pages 24–46. Springer, 2024. doi:10.1007/978-3-031-57231-9_2.

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

12 Sandra Alves, Delia Kesner, and Daniel Ventura. A quantitative understanding of pattern matching, 2019. arXiv:1912.01914.

13 Victor Arrial, Giulio Guerrieri, and Delia Kesner. Quantitative Inhabitation for Different Lambda Calculi in a Unifying Framework. *Proceedings of the ACM on Programming Languages*, 7(POPL):51:1483–51:1513, January 2023. doi:10.1145/3571244.

14 Victor Arrial, Giulio Guerrieri, and Delia Kesner. The benefits of diligence. In Chris Benzmüller, Marijn Heule, and Renate Schmidt, editors, *12th International Joint Conference on Automated Reasoning (IJCAR), 2024, Proceedings*, Lecture Notes in Artificial Intelligence. Springer, 2024.

15 Victor Arrial, Giulio Guerrieri, and Delia Kesner. Genericity through stratification. *CoRR*, abs/2401.12212, 2024. doi:10.48550/arXiv.2401.12212.

16 Davide Barbarossa and Giulio Manzonetto. Taylor subsumes scott, berry, kahn and plotkin. *Proc. ACM Program. Lang.*, 4(POPL):1:1–1:23, 2020. doi:10.1145/3371069.

17 Henk Barendregt. *Some extensional term models for combinatory logics and λ-calculi*. PhD thesis, Univ. Utrecht, January 1971.

18 Henk Barendregt. A characterization of terms of the lambda i-calculus having a normal form. *Journal Symbolic Logic*, 38(3):441–445, 1973. doi:10.2307/2273041.

19 Henk Barendregt. Solvability in lambda-calculi. In M. Guillaume, editor, *Colloque international de logique: Clermont-Ferrand, 18-25 juillet 1975*, pages 209–219, Paris, 1977. Éditions du CNRS.

20 Henk Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in logic and the foundation of mathematics*. North-Holland, Amsterdam, revised edition, 1984.

21 Jan Bessai, Tzu-Chun Chen, Andrej Dudenhefner, Boris Düdder, Ugo de'Liguoro, and Jakob Rehof. Mixin composition synthesis based on intersection types. *CoRR*, abs/1712.06906, 2017. arXiv:1712.06906.

22 Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. Proving soundness of extensional normal-form bisimilarities. *Log. Methods Comput. Sci.*, 15(1), 2019. doi:10.23638/LMCS-15(1:31)2019.

23 Antonio Bucciarelli, Delia Kesner, Alejandro Ríos, and Andrés Viso. The bang calculus revisited. In *Functional and Logic Programming - 15th International Symposium, FLOPS 2020, Proceedings*, volume 12073 of *Lecture Notes in Computer Science*, pages 13–32. Springer, 2020. doi:10.1007/978-3-030-59025-3_2.

24 Antonio Bucciarelli, Delia Kesner, Alejandro Ríos, and Andrés Viso. The bang calculus revisited. *Inf. Comput.*, 293:105047, 2023. doi:10.1016/J.IC.2023.105047.

25 Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. The inhabitation problem for non-idempotent intersection types. In Josep Díaz, Ivan Lanese, and Davide Sangiorgi, editors, *Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*, volume 8705 of *Lecture Notes in Computer Science*, pages 341–354. Springer, 2014. `doi:10.1007/978-3-662-44602-7_26`.

26 Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. Solvability = Typability + Inhabitation. *Logical Methods in Computer Science*, Volume 17, Issue 1, January 2021. `doi:10.23638/LMCS-17(1:7)2021`.

27 Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. Solvability = typability + inhabitation. *Logical Methods in Computer Science*, 17(1), 2021. URL: `https://lmcs.episciences.org/7141`.

28 Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. Inhabitation for non-idempotent intersection types. *Logical Methods in Computer Science*, 14(3), 2018. `doi:10.23638/LMCS-14(3:7)2018`.

29 Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. Observability for pair pattern calculi. In *TLCA*, volume 38 of *LIPIcs*, pages 123–137. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, 2015.

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

31 Mario Coppo and Mariangiola Dezani-Ciancaglini. A new type assignment for λ-terms. *Archiv für mathematische Logik und Grundlagenforschung*, 19(1):139–156, 1978. `doi:10.1007/BF02011875`.

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

33 Mario Coppo, Mariangiola Dezani-Ciancaglini, and Betti Venneri. Functional characters of solvable terms. *Mathematical Logic Quarterly*, 27(2-6):45–58, 1981. `doi:10.1002/malq.19810270205`.

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

35 Pierre-Louis Curien and Guillaume Munch-Maccagnoni. The duality of computation under focus. In Cristian S. Calude and Vladimiro Sassone, editors, *Theoretical Computer Science - 6th IFIP TC 1/WG 2.2 International Conference, TCS 2010, Held as Part of WCC 2010. Proceedings*, volume 323 of *IFIP Advances in Information and Communication Technology*, pages 165–181. Springer, 2010. `doi:10.1007/978-3-642-15240-5_13`.

36 Daniel de Carvalho. *Sémantiques de la logique linéaire et temps de calcul*. PhD thesis, Université Aix-Marseille II, 2007.

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

38 Thomas Ehrhard. Call-by-push-value from a linear logic point of view. In Peter Thiemann, editor, *Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Proceedings*, volume 9632 of *Lecture Notes in Computer Science*, pages 202–228, Eindhoven, The Netherlands, 2016. Springer. `doi:10.1007/978-3-662-49498-1_9`.

39 Thomas Ehrhard and Giulio Guerrieri. The bang calculus: an untyped lambda-calculus generalizing call-by-name and call-by-value. In James Cheney and Germán Vidal, editors, *Proceedings of the 18th International Symposium on Principles and Practice of Declarative Programming*, pages 174–187, Edinburgh, United Kingdom, 2016. ACM. doi:10.1145/2967973.2968608.

40 José Espírito Santo, Luís Pinto, and Tarmo Uustalu. Modal embeddings and calling paradigms. In *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019*, volume 131 of *LIPIcs*, pages 18:1–18:20. Schloss Dagstuhl, 2019. doi:10.4230/LIPIcs.FSCD.2019.18.

41 Claudia Faggian and Giulio Guerrieri. Factorization in call-by-name and call-by-value calculi via linear logic. In *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Proceedings*, volume 12650 of *Lecture Notes in Computer Science*, pages 205–225. Springer, 2021. doi:10.1007/978-3-030-71995-1_11.

42 Álvaro García-Pérez and Pablo Nogueira. No solvable lambda-value term left behind. *Logical Methods in Computer Science*, 12(2):1–43, 2016. doi:10.2168/LMCS-12(2:12)2016.

43 Philippa Gardner. Discovering needed reductions using type theory. In Masami Hagiya and John C. Mitchell, editors, *Theoretical Aspects of Computer Software*, pages 555–574, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

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

45 Jean-Yves Girard. On the unity of logic. *Ann. Pure Appl. Log.*, 59(3):201–217, 1993. doi:10.1016/0168-0072(93)90093-S.

46 Giulio Guerrieri and Giulio Manzonetto. The bang calculus and the two girard's translations. In *Proceedings Joint International Workshop on Linearity & Trends in Linear Logic and Applications, Linearity-TLLA@FLoC 2018*, volume 292 of *EPTCS*, pages 15–30, Oxford, UK, 2018. doi:10.4204/EPTCS.292.2.

47 Giulio Guerrieri, Luca Paolini, and Simona Ronchi Della Rocca. Standardization and conservativity of a refined call-by-value lambda-calculus. *Logical Methods Computer Science*, 13(4):1–27, 2017. doi:10.23638/LMCS-13(4:29)2017.

48 Richard Kennaway, Vincent van Oostrom, and Fer-Jan de Vries. Meaningless terms in rewriting. *J. Funct. Log. Program.*, 1999(1), 1999. URL: http://danae.uni-muenster.de/lehre/kuchen/JFLP/articles/1999/A99-01/A99-01.html.

49 Axel Kerinec, Giulio Manzonetto, and Michele Pagani. Revisiting call-by-value böhm trees in light of their taylor expansion. *Logical Methods in Computer Science*, 16(3):6:1–6:26, 2020. URL: https://lmcs.episciences.org/6638.

50 Delia Kesner, Victor Arrial, and Giulio Guerrieri. Meaningfulness and genericity in a subsuming framework. *CoRR*, abs/2404.06361, 2024. URL: https://arxiv.org/abs/2404.06361.

51 Delia Kesner and Shane Ó Conchúir. Milner's lambda-calculus with partial substitutions. *CoRR*, abs/2312.13270, 2023. doi:10.48550/arXiv.2312.13270.

52 Delia Kesner and Andrés Viso. Encoding tight typing in a unified framework. In *30th EACSL Annual Conference on Computer Science Logic, CSL 2022*, volume 216 of *LIPIcs*, pages 27:1–27:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.CSL.2022.27.

53 Søren B. Lassen. Eager normal form bisimulation. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*, pages 345–354. IEEE Computer Society, 2005. doi:10.1109/LICS.2005.15.

54 Paul Blain Levy. Call-by-push-value: A subsuming paradigm. In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications*, pages 228–243, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. doi:10.1007/3-540-48959-2_17.

55 Paul Blain Levy. *Call-By-Push-Value: A Functional/Imperative Synthesis*, volume 2 of *Semantics Structures in Computation*. Springer, 2004.

56 Zohar Manna and Richard J. Waldinger. A deductive approach to program synthesis. *ACM Transactions on Programming Languages and Systems*, 2(1):90–121, 1980. doi:10.1145/357084.357090.

57 John Maraist, Martin Odersky, David N. Turner, and Philip Wadler. Call-by-name, call-by-value, call-by-need and the linear lambda calculus. In Stephen D. Brookes, Michael G. Main, Austin Melton, and Michael W. Mislove, editors, *Eleventh Annual Conference on Mathematical Foundations of Programming Semantics, MFPS 1995, Tulane University, New Orleans, LA, USA, March 29 - April 1, 1995*, volume 1 of *Electronic Notes in Theoretical Computer Science*, pages 370–392. Elsevier, 1995. doi:10.1016/S1571-0661(04)00022-2.

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

59 Robin Milner. Local bigraphs and confluence: two conjectures. In Roberto Amadio and Iain Phillips, editors, *Proceedings of the 13th Int. Workshop on Expressiveness in Concurrency (EXPRESS)*, volume 175, pages 65–73. Electronic Notes in Theoretical Computer Science, 2006. doi:10.1016/j.entcs.2006.07.035.

60 Eugenio Moggi. Computational lambda-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*, pages 14–23. IEEE Computer Society, 1989. doi:10.1109/LICS.1989.39155.

61 Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, 1991. doi:10.1016/0890-5401(91)90052-4.

62 Guillaume Munch-Maccagnoni. Focalisation and classical realisability. In *Computer Science Logic, 23rd international Workshop, CSL 2009, 18th Annual Conference of the EACSL. Proceedings*, volume 5771 of *Lecture Notes in Computer Science*, pages 409–423. Springer, 2009. doi:10.1007/978-3-642-04027-6_30.

63 Luca Paolini, Elaine Pimentel, and Simona Ronchi Della Rocca. An operational characterization of strong normalization. In Luca Aceto and Anna Ingólfsdóttir, editors, *Foundations of Software Science and Computation Structures, 9th International Conference, FOSSACS 2006, Proceedings*, volume 3921 of *Lecture Notes in Computer Science*, pages 367–381, Vienna, Austria, 2006. Springer. doi:10.1007/11690634_25.

64 Luca Paolini and Simona Ronchi Della Rocca. Call-by-value solvability. *RAIRO Theoretical Informatics and Applications*, 33(6):507–534, 1999. doi:10.1051/ita:1999130.

65 Daniele Pautasso and Simona Ronchi Della Rocca. A quantitative version of simple types. In Marco Gaboardi and Femke van Raamsdonk, editors, *8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023, July 3-6, 2023, Rome, Italy*, volume 260 of *LIPIcs*, pages 29:1–29:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.FSCD.2023.29.

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

67 Garrel Pottinger. A type assignment for the strongly normalizable λ-terms. In *To H.B.Curry: Essays on Combinatory Logic, Lambda-calculus and Formalism*, pages 561–577. Academic Press, 1980.

68 Dag Prawitz. *Classical versus intuitionistic logic. Why is this a Proof?*, volume 27. College Publications, 2017.

69 Miguel Ramos. Embedding the distant bang calculus with pattern-matching primitives, 2024. Personal Communication.

70 Laurent Regnier. Une équivalence sur les lambda-termes. *Theoretical Computer Science*, 126(2):281–292, 1994. doi:10.1016/0304-3975(94)90012-4.

71 Simona Ronchi Della Rocca and Luca Paolini. *The Parametric Lambda Calculus - A Metamodel for Computation*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004. doi:10.1007/978-3-662-10394-4.

72 Simona Ronchi Della Rocca and Luca Roversi. Lambda calculus and intuitionistic linear logic. *Stud Logica*, 59(3):417–448, 1997. doi:10.1023/A:1005092630115.

73 Nicolas Wu Steffen van Bakel, Emma Tye. A calculus of delayed reductions. In *PPDP 2023: 25th International Symposium on Principles and Practice of Declarative Programming, Cascais, Lisbon, Portugal, October 22 - 23, 2022*. ACM, 2023.

74 Pawel Urzyczyn. The emptiness problem for intersection types. *Journal of Symbolic Logic*, 64(3):1195–1215, 1999. doi:10.2307/2586625.

75 Christopher P. Wadsworth. *Semantics and pragmatics of the lambda-calculus*. Ph.D., University of Oxford, 1971. Accepted: 1971. URL: https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.476186.

76 Christopher P. Wadsworth. The Relation between Computational and Denotational Properties for Scott's D_∞-Models of the Lambda-Calculus. *SIAM Journal on Computing*, 5(3):488–521, September 1976. Publisher: Society for Industrial and Applied Mathematics. doi:10.1137/0205036.