Call-By-Value, Again!

Axel Kerinec
Laboratoire LIPN, CNRS UMR 7030, Université Sorbonne Paris-Nord, France

Giulio Manzonetto
Laboratoire LIPN, CNRS UMR 7030, Université Sorbonne Paris-Nord, France

Simona Ronchi Della Rocca
Computer Science Department, University of Torino, Italy

Abstract

The quest for a fully abstract model of the call-by-value λ-calculus remains crucial in programming language theory, and constitutes an ongoing line of research. While a model enjoying this property has not been found yet, this interesting problem acts as a powerful motivation for investigating classes of models, studying the associated theories and capturing operational properties semantically.

We study a relational model presented as a relevant intersection type system, where intersection is in general non-idempotent, except for an idempotent element that is injected in the system. This model is adequate, equates many λ-terms that are indeed equivalent in the maximal observational theory, and satisfies an Approximation Theorem w.r.t. a system of approximants representing finite pieces of call-by-value Böhm trees. We show that these tools can be used for characterizing the most significant properties of the calculus – namely valuability, potential valuability and solvability – both semantically, through the notion of approximants, and logically, by means of the type assignment system. We mainly focus on the characterizations of solvability, as they constitute an original result. Finally, we prove the decidability of the inhabitation problem for our type system by exhibiting a non-deterministic algorithm, which is proven sound, correct and terminating.

2012 ACM Subject Classification Theory of computation → Denotational semantics; Theory of computation → Linear logic

Keywords and phrases λ-calculus, call-by-value, intersection types, solvability, inhabitation

Digital Object Identifier 10.4230/LIPIcs.FSCD.2021.7

Funding *Giulio Manzonetto*: This work is partly supported by ANR Project ANR-19-CE48-0014.

Acknowledgements We would like to thank Giulio Guerrieri for interesting discussions, as well as the anonymous reviewers for helpful suggestions.

Introduction

Despite the fact that the call-by-value (CbV) λ-calculus has been introduced by Plotkin several decades ago [22], the problem of finding a denotational model satisfactorily reflecting its operational semantics is not completely solved, yet. While a plethora of adequate models has been constructed, e.g., in the Scott continuous and stable semantics [13, 23, 18], none enjoys completeness and it is therefore fully abstract. Similarly, the theory of program approximations for the CbV λ-calculus remained for a longtime rather involuted compared to the one developed in the call-by-name (CbN) setting (see [5, Ch. 14]). As an example, in [26] the authors show that the continuous model built in [13] does satisfy an Approximation Theorem, but the considered notion of approximant turns out to be too weak for capturing any interesting operational property. The main problem one encounters when approximating CbV reductions is that certain redexes remain stuck along reductions for silly reasons, thus preventing the creation of other redexes and leading to premature CbV normal forms (see [2]). A possible solution has been proposed in [10] by introducing permutation reductions that allow to unblock such redexes without altering fundamental operational properties of the

© Axel Kerinec, Giulio Manzonetto, and Simona Ronchi Della Rocca;
licensed under Creative Commons License CC-BY 4.0

calculus, like the capability of a program of reducing to a value or the notion of solvability (as shown in [17]). This breakthrough has renewed the interest in the CbV λ-calculus within the scientific community and led to a wealth of original results [2, 17, 3, 20, 19].

Inspired by the relational semantics of Linear Logic [15] and exploiting Girard's "boring" translation of intuitionistic arrow in linear logic, sending $A \to B$ into $!(A \multimap B)$, Ehrhard introduced a class of relational models for CbV λ-calculus [14]. Like filter models correspond to intersection type systems under the celebrated Stone duality [1], also relational models can be nicely presented in a similar fashion [25], except for the fact that the intersection becomes a non-idempotent operator. Thus, the type $\alpha_1 \wedge \cdots \wedge \alpha_n$ can be seen as a multiset $[\alpha_1, \ldots, \alpha_n]$. The advantage of counting the multiplicities is that it allows to expose quantitative properties of programs, e.g., by extracting a bound to their head reduction sequences [12]. The disadvantage of using relational models is that they are extremely poor in terms of representable theories. In the CbN setting, it is clear from [7] that all non-extensional relational graph models induce the same theory, and we have reasons to believe that the same holds for the class of CbV models from [14]. Therefore, in order to obtain different theories, one needs to substantially modify the construction of the model. Now, in coherent spaces, it is possible construct CbV models by performing a "lifting" that injects a new point $\star$ (coherent with all existing points) [18], leading to a solution of the domain equation $\mathcal{D} \cong [\mathcal{D} \to_s \mathcal{D}] \oplus \{\star\}$, where $[\cdot \to_s \cdot]$ denotes the domain of stable functions [6]. Mimicking this construction in the relational semantics, a new class of relational models for the CbV λ-calculus was introduced in [20]. The main difference is that, in the associated type assignment systems, the intersection is still non-idempotent except for an idempotent element $[]$, which is available at will and can be used to type any value in the empty environment. The authors show that all models in this class satisfy adequacy, a property they share with Ehrhard's relational models, but induce different theories. More precisely, they equate many λ-terms that are indeed equal in the maximal observational equivalence, whence the induced theory is closer to full abstraction. A notable example is given by the λ-terms $(\lambda x.xx)M$ and MM that are observationally indistinguishable – even when M is not a value – but have distinct interpretations in Ehrhard's models [16]. Moreover it has been proved that all models in this class enjoy an Approximation Theorem.

In this paper we study a particular relational model $\mathcal{M}$ living in the class of [20], corresponding to a relevant intersection type system having countably many atoms and no additional equivalences among types (in particular, atoms are not equivalent to arrow types). We show that the model $\mathcal{M}$ satisfies an Approximation Theorem with respect to a refined notion of syntactic approximants, that take permutation rules into account and were successfully applied in [19] to introduce a CbV notion of Böhm trees. By exploiting the resource consciousness of the model, we are able to provide an easy inductive proof of this result (Theorem 23) and avoid the impredicative techniques based on reducibility candidates that are needed in the continuous and stable semantics (see, e.g., [4, Ch. 17] or [26, Thm. 11.1.19]). As a consequence of the Approximation Theorem we derive that the model $\mathcal{M}$ equates all λ-terms having the same CbV Böhm tree. The fact that $\mathcal{M}$ is sensitive to the amount of resources needed by a λ-term during its execution still breaks the full abstraction property (a counter-example is given in [20]).

Despite the lack of full abstraction, we show that the model $\mathcal{M}$ and the associated system of approximants allow to characterize nicely operational properties like *valuability*, *potential valuability*, and *solvability*. A λ-term is (potentially) valuable if it reduces to a value (under suitable substitutions), and solvable if it is capable of generating a completely defined result, like the identity, when plugged in a suitable context. The notion of solvability, inherited from the CbN λ-calculus, is particularly interesting since it identifies the "meaningful" programs. In CbN, solvability has been characterized operationally (via head reduction), logically (through typability) and semantically (by building models assigning non-trivial interpretations to solvable terms exclusively). The model $\mathcal{M}$ provides a logical and semantic characterization of CbV solvability. On the logical side, we show that a λ-term is solvable if and only if it is typable in $\mathcal{M}$ with types that are "proper", in the sense of Definition 31. On the semantic side, we prove that all solvables admit approximants having a particular shape. Both the logical and semantic characterizations are presented in Theorem 36 and, as a consequence, we obtain that $\mathcal{M}$ is not sensible, but semi-sensible. This means that the model is "meaningful" since it does not equate all unsolvables, but neither equates a solvable to an unsolvable.

Finally, since the model $\mathcal{M}$ is presented as an intersection type system, it feels natural to wonder whether the type inhabitation problem is decidable.

> The Inhabitation Problem (IHP): Given any type environment Γ and any type α, is there a λ-term M having type α in Γ?

Since Urzyczyn's work [27], it is known that IHP is undecidable for the CbN (idempotent) intersection type system presented in [11]. Van Bakel subsequently simplified the system using strict types [29], where intersection is only allowed on the left-hand side of an arrow, while maintaining the undecidability of inhabitation – even in its "relevant" version where type environments only contain the consumed premises (Urzyczyn's proof extends easily [28]). On the one hand, this shows that the decidability of inhabitation is not strictly connected with the relevance of the system, on the other hand IHP has been proven decidable for several non-idempotent intersection type systems [9]. In Section 4, we describe a non-deterministic algorithm taking an environment Γ and a type α as inputs, and generating as output all minimal approximants having type α in Γ. First, we show that the algorithm is terminating (Theorem 46), a result only possible because there are finitely many approximants satisfying the above criteria. Then we demonstrate the soundness and completeness properties of the algorithm (Theorem 48), from which the decidability of IHP in this setting follows. Although our inhabitation algorithm is clearly inspired by [9], the adaptation is non-trivial for two reasons: the presence in the CbV setting of normal inhabitants having the shape $(\lambda x.M)N$ of a β-redex, and the presence of an idempotent element in the type assignment system.

Some related works

Despite the existence of several models of the CbV λ-calculus, their theories have rarely been explored. An exception is [26], where the theory of the model from [13] has been extensively studied. A semantic characterization of solvability is given, but not completely satisfactory because of the weak notion of approximation employed. The first logical characterization of CbV solvability is in [21], through a particular class of (idempotent) intersection types – it is, in some sense, similar to ours, but it is not based on a semantic model. Two known attempts at characterizing this notion from an operational point of view are [21, 10], both based on *ad hoc* reduction rules that are however unsound for CbV semantics. This suggests that CbV languages still lack a satisfactory rewriting theory.

1 Preliminaries

For the syntax of λ-calculus we mainly follow Barendregt's first book [5], for its call-by-value version [26], and for its extension with permutation rules [10].

1.1 The call-by-value λ-calculus

We consider fixed a countable set $\mathbb{V}$ of variables. The set Λ of λ-*terms* and the set Val of *values* are defined inductively via the following grammar (where $x \in \mathbb{V}$):

$$(\Lambda)\quad M, N ::= MN \mid V \qquad\qquad (\mathrm{Val})\quad V, U ::= x \mid \lambda x.M$$

Application is represented as juxtaposition. As usual, we assume that it associates to the left and has higher-precedence than abstraction. Given $M \in \Lambda$, we shorten $\lambda x_1.(\cdots(\lambda x_k.M)\cdots)$ as $\lambda x_1 \ldots x_k.M$ or even as $\lambda\vec{x}.M$. For example, $\lambda xyz.xyz$ stands for $\lambda x.(\lambda y.(\lambda z.(xy)z))$. Given $N_1, \ldots, N_n \in \Lambda$, we write $M\vec{N}$ for $MN_1 \cdots N_n$ and $MN_1^{\sim k}$ for $MN_1 \cdots N_1$ (k-times).

The set $\mathrm{FV}(M)$ of *free variables of* M and α-*conversion* are defined as usual [5, §2.1]. We say that a λ-term M is *closed*, or *a combinator*, whenever $\mathrm{FV}(M) = \emptyset$. We denote by Λ^o the set of all combinators. From now on, λ-terms are considered up to α-conversion.

Concerning specific combinators, we fix the following notations (for $n \in \mathbb{N}$):

$$\begin{array}{lcllcllcll}
\mathbf{K} & = & \lambda xy.x & \mathbf{\Delta} & = & \lambda x.xx, & \mathbf{\Omega} & = & \mathbf{\Delta\Delta}, & \mathbf{P}_n = \lambda x_0 \ldots x_n.x_n, \\
\mathbf{B} & = & \lambda fgx.f(gx), & \mathbf{K}^\star & = & \mathbf{ZK}, & \mathbf{Z} & = & \lambda f.(\lambda y.f(\lambda z.yyz))(\lambda y.f(\lambda z.yyz)), &
\end{array}$$

where $\mathbf{K}$ is the first projection, $\mathbf{\Delta}$ the self-application, $\mathbf{\Omega}$ the paradigmatic looping combinator, $\mathbf{P}_n$ erases n arguments, $\mathbf{B}$ is the composition, $\mathbf{K}^\star$ an ogre and $\mathbf{Z}$ a CbV recursion operator. Notice that $\mathbf{P}_0 = \lambda x_0.x_0$ is the identity, therefore we also use $\mathbf{I}$ as an alternative notation.

▶ **Definition 1.** *On Λ, we define the following notions of reduction (for $V \in \mathrm{Val}$):*

$$\begin{array}{llll}
(\beta_{\mathsf{v}}) & (\lambda x.M)V & \to & M[V/x], \quad \textit{where } [V/x] \textit{ denotes capture-free substitution,} \\
(\sigma_1) & (\lambda x.M)NP & \to & (\lambda x.MP)N, \quad \textit{with } x \notin \mathrm{FV}(P), \\
(\sigma_3) & V((\lambda x.M)N) & \to & (\lambda x.VM)N, \quad \textit{with } x \notin \mathrm{FV}(V).
\end{array}$$

We also define $(\sigma) = (\sigma_1) \cup (\sigma_3)$ and $(\mathsf{v}) = (\beta_{\mathsf{v}}) \cup (\sigma)$. Each $\mathsf{R} \in \{\beta_{\mathsf{v}}, \sigma_1, \sigma_3, \sigma, \mathsf{v}\}$ induces a one-step *(resp.* multi-steps*)* reduction *relation $\to_{\mathsf{R}}$ ($\twoheadrightarrow_{\mathsf{R}}$), and a* conversion *relation $=_{\mathsf{R}}$. We say that a λ-term M* is in R-normal form *(R-nf, for short) if there is no $N \in \Lambda$ such that $M \to_{\mathsf{R}} N$. We say that M* has an R-nf *if $M \twoheadrightarrow_{\mathsf{R}} N$ for some λ-term N in R-nf.*

▶ **Fact 2.** *The set* Val *is closed under substitutions $\vartheta : \mathbb{V} \to \mathrm{Val}$ and* v*-reductions.*

Plotkin's original formulation of the CbV λ-calculus only considers the β_{v}-reduction [22]. The permutation rules (σ), introduced by Regnier in the CbN setting [24], have been extended by Carraro and Guerrieri to CbV in [10], where the following properties are shown.

▶ **Proposition 3.**

(i) *The reduction $\to_\sigma$ is strongly normalizing. More precisely, there exists a measure $\mathsf{s} : \Lambda \to \mathbb{N}$ such that $M \to_\sigma N$ entails $\mathsf{s}(N) < \mathsf{s}(M)$.*

(ii) *The reduction $\to_{\mathsf{v}}$ is confluent. In particular, the* v*-nf of $M \in \Lambda$ (if any) is unique.*

▶ **Example 4.**

(i) $\mathbf{\Omega} \to_{\beta_{\mathsf{v}}} \mathbf{\Omega}$, $\lambda x.\mathbf{\Omega} \to_{\beta_{\mathsf{v}}} \lambda x.\mathbf{\Omega}$ and $\mathbf{I}x \to_{\beta_{\mathsf{v}}} x$, while $\mathbf{I}(xy)$ is a v-nf.

(ii) $(\lambda y.\mathbf{\Delta})(x\mathbf{I})\mathbf{\Delta}$ is a β_{v}-nf, but $(\lambda y.\mathbf{\Delta})(x\mathbf{I})\mathbf{\Delta} \to_{\sigma_1} (\lambda y.\mathbf{\Omega})(x\mathbf{I}) \to_{\beta_{\mathsf{v}}} (\lambda y.\mathbf{\Omega})(x\mathbf{I}) \to_{\beta_{\mathsf{v}}} \cdots$

(iii) $\mathbf{I}(\mathbf{\Delta}(xx))$ is a β_{v}-nf, but contains a σ_3-redex, indeed $\mathbf{I}(\mathbf{\Delta}(xx)) \to_{\sigma_3} (\lambda z.\mathbf{I}(zz))(xx)$.

(iv) $\mathbf{Z}$ is called a recursion operator since $\mathbf{Z}V =_{\beta_{\mathsf{v}}} V(\lambda x.\mathbf{Z}Vx)$, for all $V \in \mathrm{Val}$ and x fresh.

(v) $\mathbf{K}^\star =_{\mathsf{v}} \mathbf{K}(\lambda y.\mathbf{K}^\star y) =_{\mathsf{v}} \lambda x_0x_1.\mathbf{K}^\star x_1 =_{\mathsf{v}} \lambda x_0x_1x_2.\mathbf{K}^\star x_2 =_{\mathsf{v}} \cdots =_{\mathsf{v}} \lambda x_0 \ldots x_n.\mathbf{K}^\star x_n$.

(vi) For all $\vec{V} \in \mathrm{Val}$, we have $\mathbf{K}^\star\vec{V} =_{\mathsf{v}} \mathbf{K}^\star$ and $\mathbf{P}_nV_1 \cdots V_m \twoheadrightarrow_{\mathsf{v}} \mathbf{P}_{n-m}$ provided $n \geq m$.

Lambda terms are classified into valuable, potentially valuable, solvable or unsolvable depending on their behavior and their capability of interaction with the environment.

▶ **Definition 5.** *A λ-term M is called:*

(i) valuable *if it reduces to a value, namely $M \twoheadrightarrow_{\mathsf{v}} V$ for some $V \in \mathrm{Val}$.*
(ii) potentially valuable *if there exists a substitution $\vartheta : \mathbb{V} \to \mathrm{Val}$ such that M^{ϑ} is valuable.*
(iii) solvable *if there exist sequences $\vec{x}, \vec{V} \in \mathrm{Val}$ such that $(\lambda\vec{x}.M)\vec{V} \twoheadrightarrow_{\mathsf{v}} \mathbf{I}$.*
(iv) unsolvable*, if it is not solvable.*

Notice that the notions of solvability and valuability are both stronger than potential valuability, but orthogonal with each other. We provide some discriminating examples.

▶ **Example 6.**

(i) $\mathbf{I}, \mathbf{\Delta}, \mathbf{P}_n, \mathbf{\Delta}(\mathbf{II}), \mathbf{P}_1(\lambda x.\mathbf{\Omega})$ are (potentially) valuable and solvable.
(ii) $\mathbf{P}_1 x(\lambda x.\mathbf{\Omega}), xy\mathbf{I\Delta}$ and $\mathbf{\Delta}(xy)$ are not valuable, but potentially valuable and solvable.
(iii) $\lambda x.\mathbf{\Omega}, \mathbf{ZB}$ and $\mathbf{K}^{\star}$ are valuable, but unsolvable. The term $\mathbf{K}^{\star}$ is called an ogre because of its capability of eating any $\vec{V}$ while remaining valuable: $\mathbf{K}^{\star}\vec{V} \twoheadrightarrow_{\beta_{\mathsf{v}}} \lambda x.M \in \mathrm{Val}$.
(iv) $\mathbf{\Omega}, \mathbf{\Omega}(xy), (\lambda y.\mathbf{\Delta})(x\mathbf{I})\mathbf{\Delta}, \mathbf{I\Omega}, \mathbf{ZI}$ are not potentially valuable nor solvable. The same holds for $\mathbf{Y}M$, where $\mathbf{Y}$ is a CbN fixed point operator and M is a λ-term.

▶ Remark 7. The original definitions of valuability, potential valuability and solvability are given in terms of β_{v}-reduction (see [22] and [26], respectively). In [10] and [17], it is shown that all these notions are preserved when considering the extended v-reduction. In particular, for all λ-terms M, we have that $M \twoheadrightarrow_{\beta_{\mathsf{v}}} \mathbf{I}$ holds exactly when $M \twoheadrightarrow_{\mathsf{v}} \mathbf{I}$ does.

▶ **Property 8.** *If $M = (\lambda x_1 \dots x_k.P)N_1 \cdots N_n \twoheadrightarrow_{\mathsf{v}} \mathbf{I}$ then each N_i is valuable, say $N_i \twoheadrightarrow_{\mathsf{v}} V_i$. Moreover, we must have $k \leq n + 1$.*

Proof. By the above remark, $M \twoheadrightarrow_{\mathsf{v}} \mathbf{I}$ entails $M \twoheadrightarrow_{\beta_{\mathsf{v}}} \mathbf{I}$. Therefore, $k > n + 1$ would imply $M \twoheadrightarrow_{\beta_{\mathsf{v}}} (\lambda\vec{x}.P)\vec{V} \twoheadrightarrow_{\beta_{\mathsf{v}}} \lambda x_{n+1} \dots x_k.P' \neq_{\beta_{\mathsf{v}}} \mathbf{I}$. ◀

For the model theory of CbV λ-calculus, we refer to [26]. Every model $\mathcal{S}$ comes equipped with an interpretation map $[\![-]\!]$ that allows to compute the *denotation* $[\![M]\!]$ of $M \in \Lambda$. We say that $\mathcal{S}$ *equates* $M, N \in \Lambda$ whenever $[\![M]\!] = [\![N]\!]$. The least requirement for a model $\mathcal{S}$ is that it equates all β_{v}-convertible λ-terms (*soundness*). A model $\mathcal{S}$ is called *consistent* if it does not equate all λ-terms; *inconsistent* if it is not consistent; *sensible* if it is consistent and equates all unsolvables; *semi-sensible* if it does not equate a solvable and an unsolvable.

2 A Call-by-Value Relational Model

We define a particular model $\mathcal{M}$ living in the class of relational models introduced in [20]. A model $\mathcal{S}$ in this class can be described as a type assignment system, where finite multisets of types appear at the left-hand side of an arrow. Such a model $\mathcal{S}$ is uniquely identified by a set $\mathbb{A}$ of atomic types and a congruence $\simeq$ on types, respecting the multiset cardinalities. The model $\mathcal{M}$ under consideration corresponds to the relational model having countably many atoms, and the trivial congruence relation on types (namely, $\simeq$ is the equality $=$).

2.1 The Type Assignment System $\mathcal{M}$

In order to define the type assignment system $\mathcal{M}$, we need to introduce some basic notions and notations concerning finite multisets. Given a set A, we represent a finite multiset over A as an unordered list $[\alpha_1, \dots, \alpha_n]$, possibly with repetitions, where $n \in \mathbb{N}$ and each $\alpha_i \in A$.

$$\frac{}{x : [\alpha] \vdash x : \alpha}\ (\mathrm{var}) \qquad \frac{\Gamma, x : \sigma \vdash M : \alpha}{\Gamma \vdash \lambda x.M : \sigma \to \alpha}\ (\mathrm{lam}) \qquad \frac{\Gamma_0 \vdash M : \sigma \to \alpha \quad \Gamma_1 \vdash N : \sigma}{\Gamma_0 + \Gamma_1 \vdash MN : \alpha}\ (\mathrm{app})$$

$$\frac{\Gamma_1 \vdash M : \alpha_1 \quad \cdots \quad \Gamma_n \vdash M : \alpha_n \quad n > 0}{\sum_{i=1}^{n} \Gamma_i \vdash M : [\alpha_1, \dots, \alpha_n]}\ (\mathrm{val}_{>0}) \qquad \frac{V \in \mathrm{Val}}{\vdash V : []}\ (\mathrm{val}_0)$$

Figure 1 The inference rules of the type assignment system $\mathcal{M}$. In (lam) we assume $x \notin \mathrm{dom}(\Gamma)$.

The empty multiset will be denoted by $[]$. We write $\mathcal{M}_{\mathrm{f}}(A)$ for the set of all finite multisets over A. Given $\sigma, \tau \in \mathcal{M}_{\mathrm{f}}(A)$, we write $\sigma + \tau$ for their multiset union. The operator $+$ extends to the n-ary case $\sigma_1, \dots, \sigma_n \in \mathcal{M}_{\mathrm{f}}(A)$ in the obvious way, in symbols, $\sum_{i=1}^{n} \sigma_i \in \mathcal{M}_{\mathrm{f}}(A)$.

▶ **Definition 9.** *Let us fix a countable set* $\mathbb{A} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \dots\}$ *of constants called* atomic types.

(i) *The set* $\mathbb{T}$ *of* types *over* $\mathbb{A}$ *and the set* $\mathbb{T}^{!}$ *of* multiset types *are defined by (for* $n \geq 0$*):*

$$\begin{array}{llll} (\mathbb{T}) & \alpha, \beta & ::= & \mathsf{a} \mid [] \mid \sigma \to \alpha \\ (\mathbb{T}^{!}) & \sigma, \tau, \rho & ::= & [\alpha_1, \dots, \alpha_n] \quad \textit{with } \alpha_i \neq [], \textit{ for all } i\,(1 \leq i \leq n). \end{array}$$

The arrow is right associative, i.e., $\sigma_1 \to \cdots \to \sigma_n \to \alpha = (\sigma_1 \to (\cdots (\sigma_n \to \alpha) \cdots))$.

(ii) Type environments *are functions* $\Gamma : \mathbb{V} \to \mathbb{T}^{!}$ *having a finite domain, which is defined by* $\mathrm{dom}(\Gamma) = \{x \mid \Gamma(x) \neq []\}$*. The multiset sum is extended to type environments* Γ *and* Δ *pointwisely, namely, by setting* $(\Gamma + \Delta)(x) = \Gamma(x) + \Delta(x)$*, for all* $x \in \mathbb{V}$.

(iii) *We denote by* $x_1 : \sigma_1, \dots, x_n : \sigma_n$ *the type environment* Γ *defined by setting:*

$$\Gamma(x) = \begin{cases} \sigma_i, & \textit{if } y = x_i \textit{ for some } i \in \{1, \dots, n\}, \\ [], & \textit{otherwise.} \end{cases}$$

Intuitively, the multiset type $[\alpha_1, \dots, \alpha_n] \in \mathbb{T}^{!} = \mathcal{M}_{\mathrm{f}}(\mathbb{T} - \{[]\})$ represents an intersection type $\alpha_1 \wedge \cdots \wedge \alpha_n$, where $\wedge$ enjoys associativity and commutativity, but not idempotency ($\alpha \wedge \alpha \neq \alpha$). The empty multiset $[]$ belongs both to $\mathbb{T}$ and $\mathbb{T}^{!}$, but with different meanings: $[] \in \mathbb{T}$ should be thought of as a special "idempotent" type atom which is available at will; morally, $[] \in \mathbb{T}^{!}$ is a multiset only containing an indeterminate amount of atoms $[] \in \mathbb{T}$.

▶ **Definition 10.**

(i) *A* typing judgement *has shape* $\Gamma \vdash M : \xi$*, where* Γ *is an environment,* $M \in \Lambda$ *and* $\xi \in \mathbb{T} \cup \mathbb{T}^{!}$*. The inference rules of the type system* $\mathcal{M}$ *are given in Figure 1.*

(ii) *We write* $\Pi \ \triangleright \Gamma \vdash M : \xi$ *to indicate that* Π *is a derivation of* $\Gamma \vdash M : \xi$*. Hereafter, when writing* $\Gamma \vdash M : \xi$*, we assume that* $\Pi \ \triangleright \Gamma \vdash M : \xi$ *holds for some derivation* Π.

The rules (var), (lam) and (app) are self-explanatory. In case $x \notin \mathrm{FV}(M)$, the rule (lam) assigns $\lambda x.M$ the type $[] \to \alpha$ in the environment Γ. The rule (val_0) can be used to type every value with $[]$ in the empty environment. The rule $(\mathrm{val}_{>0})$ allows to collect several types of M into a non-empty multiset type, by adding the corresponding environments together. The type system is relevant in the sense that $\Gamma \vdash M : \alpha$ entails $\mathrm{dom}(\Gamma) \subseteq \mathrm{FV}(M)$.

▶ **Example 11.** The following is a derivation Π in system $\mathcal{M}$ (setting $\Gamma = f : [[\mathsf{a}, \mathsf{a}] \to \mathsf{a}]$):

$$\dfrac{\dfrac{}{\Gamma \vdash f : [\mathsf{a}, \mathsf{a}] \to \mathsf{a}} \qquad \dfrac{\dfrac{x : [[] \to \mathsf{a}] \vdash x : [] \to \mathsf{a} \quad \vdash y : []}{x : [[] \to \mathsf{a}] \vdash xy : \mathsf{a}} \qquad \dfrac{x : [[\mathsf{b}] \to \mathsf{a}] \vdash x : [\mathsf{b}] \to \mathsf{a} \quad y : [\mathsf{b}] \vdash y : \mathsf{b}}{x : [[\mathsf{b}] \to \mathsf{a}], y : [\mathsf{b}] \vdash xy : \mathsf{a}}}{x : [[] \to \mathsf{a}, [\mathsf{b}] \to \mathsf{a}], y : [\mathsf{b}] \vdash xy : [\mathsf{a}, \mathsf{a}]}}{\Gamma + x : [[] \to \mathsf{a}, [\mathsf{b}] \to \mathsf{a}], y : [\mathsf{b}] \vdash f(xy) : \mathsf{a}}$$

Other derivable typing judgements are $\vdash \mathbf{I} : [\mathsf{a}] \to \mathsf{a}$, $\vdash \mathbf{I}x : []$ and $x : [\mathsf{a}] \vdash (\lambda y.x)x : \mathsf{a}$.

Through the rule (val_0), it is possible to assign the type $[]$ to a value V without inspecting its shape and typing its subterms[1] – we say that such a V is not *fully typed*. Similarly, in a derivation of $\Gamma \vdash M : \alpha$, certain subterms of M might not be fully typed. E.g., in any derivation of $x : [\mathsf{a}] \vdash (\lambda y.x)x : \mathsf{a}$, the former occurrence of x must be fully typed, while the latter cannot be. To identify occurrences of a subterm and formalize this intuitive property, we introduce single-hole contexts. A *single-hole context* $C[]$ is a λ-term containing exactly one occurrence of a distinguished algebraic variable $[]$, traditionally called its *hole*. Given a single-hole context $C[]$ and $N \in \Lambda$, we write $C[N]$ for the λ-term obtained by substituting N for the occurrence of the hole $[]$ in $C[]$, possibly with capture of free variables. Every such context $C[]$ uniquely identifies one occurrence of a subterm N of M, as in $M = C[N]$.

▶ **Definition 12.** *Let $M \in \Lambda$, and $\Pi \;\triangleright \Gamma \vdash M : \xi$ for some context Γ and $\xi \in \mathbb{T} \cup \mathbb{T}^!$.*

(i) *The set* $\mathsf{fto}(\Pi)$ *of* fully typed occurrences of subterms of M in Π *is the set of single-hole contexts defined by structural induction on Π and by cases on its last applied rule:*

(var) $\mathsf{fto}(\Pi) = \{[]\}$.

(lam) $\mathsf{fto}(\Pi) = \{[]\} \cup \{\lambda x.C[] \mid C[] \in \mathsf{fto}(\Pi')\}$, *if $M = \lambda x.M'$ and Π' is the premise of Π.*

(app) $\mathsf{fto}(\Pi) = \{[]\} \cup \{(C[])Q \mid C[] \in \mathsf{fto}(\Pi_1)\} \cup \{P(C[]) \mid C[] \in \mathsf{fto}(\Pi_2)\}$, *where $M = PQ$ and Π_1, Π_2 are the major and minor premises of Π, respectively.*

(val_0) $\mathsf{fto}(\Pi) = \emptyset$.

$(\mathrm{val}_{>0})$ $\mathsf{fto}(\Pi) = \bigcap_{1\le i\le n} \mathsf{fto}(\Pi'_i)$, *where $(\Pi'_i)_{1\le i\le n}$ are the premises of Π.*

(ii) *We say that N* is a typed subterm occurrence of M in Π *if $M = C[N]$ for $C[] \in \mathsf{fto}(\Pi)$.*

(iii) *On Π, define a measure $\mathsf{m}(\Pi) = \langle \mathsf{app}(\Pi), \mathsf{s}(M)\rangle \in \mathbb{N}^2$ (lexicographically ordered) where*
- *$\mathsf{app}(\Pi)$ is the number of (app) rules in Π, and*
- *$\mathsf{s}(M)$ is the measure from Proposition 3(i), strictly decreasing along (σ) steps.*

▶ Remark 13. When the last rule of Π is $(\mathrm{val}_{>0})$, a subterm occurrence is typed if and only if it is typed in all subjects of the premises. For example, in the derivation Π of Example 11, the occurrence of y in $f(xy)$ is not fully typed since $\mathsf{fto}(\Pi) = \{[], [](xy), f([]y)\}$.

▶ **Proposition 14.** *Let $M, N \in \Lambda$ be such that $M \to_{\mathsf{v}} N$, Γ be an environment and $\alpha \in \mathbb{T}$.*

(i) *(Weighted Subject Reduction) If $\Pi \;\triangleright \Gamma \vdash M : \alpha$ then $\Pi' \;\triangleright \Gamma \vdash N : \alpha$ for some Π'. Moreover, if the redex occurrence contracted in M is fully typed in Π then $\mathsf{m}(\Pi') < \mathsf{m}(\Pi)$.*

(ii) *(Subject Expansion) If $\Gamma \vdash N : \alpha$ is derivable, then so is $\Gamma \vdash M : \alpha$.*

Proof. By Lemma 4.14 in [20]. ◀

From this proposition, the soundness of the model $\mathcal{M}$ follows easily.

▶ **Definition 15.** *The* interpretation of a λ-term M in the model $\mathcal{M}$ *is given by:*

$$[\![M]\!] = \{(\Gamma, \alpha) \mid \Gamma \vdash M : \alpha\}.$$

We write $\mathcal{M} \models M = N$ whenever $[\![M]\!] = [\![N]\!]$ holds.

▶ **Corollary 16** (Soundness). *For $M, N \in \Lambda$, $M =_{\mathsf{v}} N$ entails $\mathcal{M} \models M = N$.*

[1] This includes the case $\vdash x : []$, although x contains itself as a subterm and it is assigned a type. This is consistent with the fact that (val_0) uses the information that x is a value, without looking at its shape.

2.2 The Approximation Theory of $\mathcal{M}$

We now show that the model $\mathcal{M}$ is also well-suited to model the theory of program approximation introduced in [19] for defining Call-by-Value Böhm trees. In particular, we provide a quantitative proof of the Approximation Theorem in the spirit of [7, 9, 20].

▶ **Definition 17.**

(i) *Let $\Lambda_\bot$ be the set of λ-terms possibly containing occurrences of a constant $\bot$, and* $\mathrm{Val}_\bot = \bot \cup \mathbb{V} \cup \{\lambda x.M \mid M \in \Lambda_\bot\} \subseteq \Lambda_\bot$ *the set of* extended values.

(ii) *The set $\mathcal{A}$ of* (finite) approximants *is inductively defined by the grammar (for $n \geq 0$):*

$$\begin{array}{llcl} (\mathcal{A}) & A & ::= & H \mid R \\ & H & ::= & \bot \mid x \mid \lambda x.A \mid xHA_1 \cdots A_n \\ & R & ::= & (\lambda x.A)(yHA_1 \cdots A_n) \end{array}$$

Terms of shape H are called head approximants *as they remind those used for building CbN Böhm trees, while approximants of shape R are called* redex-like *because they look like a β-redex. Let $\mathcal{H}$ (resp. $\mathcal{R}$) be the set of all head (resp. redex-like) approximants.*

(iii) *Define $\sqsubseteq_\bot \subseteq \Lambda_\bot^2$ as the least order relation compatible with abstraction and application, and including $\bot \sqsubseteq_\bot V$ for all $V \in \mathrm{Val}_\bot$. Given a set $\mathcal{X} \subseteq \Lambda_\bot$, we write $\uparrow \mathcal{X}$ if its elements are pairwise compatible, and in this case $\bigsqcup \mathcal{X}$ denotes their least upper bound.*

(iv) *For $M \in \Lambda$, define the* set $\mathcal{A}(M)$ of (finite) approximants of M *as follows*

$$\mathcal{A}(M) = \{A \in \mathcal{A} \mid \exists N \in \Lambda \,.\, M \twoheadrightarrow_{\mathsf{v}} N \text{ and } A \sqsubseteq_\bot N\}$$

We say that two λ-terms M, N have the same CbV Böhm tree *when $\mathcal{A}(M) = \mathcal{A}(N)$.*

▶ Remark 18.

(i) Although not formally needed, one could extend the v-reduction to terms in $\Lambda_\bot$ in the obvious way, and check that all approximants $A \in \mathcal{A}$ are in v-normal form. The subterm of shape H in $xHA_1 \cdots A_n$ is precisely needed to prevent a σ_3-redex.

(ii) The terminology "M and N have the same CbV Böhm tree" is consistent with [19], where the CbV Böhm tree of a λ-term M is defined as the possibly infinite tree $\bigsqcup \mathcal{A}(M)$. Indeed, it is easy to check that $\mathcal{A}(M) = \mathcal{A}(N)$ if and only if their suprema coincide.

▶ **Example 19.**

(i) $\mathcal{A}(\mathbf{\Omega}) = \mathcal{A}(\mathbf{\Omega}(xy)) = \mathcal{A}(\mathbf{ZI}) = \emptyset$.

(ii) $\mathcal{A}(\mathbf{\Delta}) = \{\bot, \lambda x.x\bot, \lambda x.xx\}$, $\mathcal{A}(\lambda x.\mathbf{\Omega}) = \{\bot\}$ and $\mathcal{A}(\mathbf{K}^*) = \{\lambda x_1 \ldots x_n.\bot \mid n \geq 0\}$.

(iii) $\mathcal{A}(\mathbf{Z}) = \bigcup_{n\in\mathbb{N}} \{\lambda f.f(\lambda z_0.f(\lambda z_1.f \cdots (\lambda z_n.f \bot Z_n) \cdots Z_1)Z_0) \mid \forall i \,.\, Z_i \in \{z_i, \bot\}\} \cup \{\bot\}$.

(iv) $\mathcal{A}(\mathbf{ZB}) = \{\bot, \lambda f_0.\bot\} \cup \{\lambda f_0 x_0.(\cdots(\lambda f_{n-1}x_{n-1}.(\lambda f_n.\bot)(f_{n-1}X_{n-1}))\cdots)(f_0 X_0) \mid$
$n > 0, \forall i \in \{1, \ldots, n\} \,.\, X_i \in \{x_i, \bot\}\}$.

▶ **Definition 20.**

(i) *The rules in Figure 1 and the interpretation $[\![-]\!]$ in Definition 15 are extended to $\Lambda_\bot$ in the obvious way. E.g.,* (val_0) *becomes $\vdash V : []$, for all $V \in \mathrm{Val}_\bot$.*

(ii) *We say that a derivation $\Pi \rhd \Gamma \vdash M : \alpha$ is in* typed v-normal form *if, for all $C[] \in \mathsf{fto}(\Pi)$, $M = C[N]$ entails N is not a v-redex.*

(iii) *A derivation Π induces a term $M_\Pi \in \Lambda_\bot$ defined by induction on Π as follows:*

$$\begin{array}{rl} (\mathrm{var}) & M_\Pi = x, \text{ if } \Pi \rhd \Gamma \vdash x : \alpha. \\ (\mathrm{lam}) & M_\Pi = \lambda x.M_{\Pi'}, \text{ if } \Pi \rhd \Gamma \vdash \lambda x.N : \alpha \text{ and } \Pi' \text{ is the premise of } \Pi. \\ (\mathrm{app}) & M_\Pi = M_{\Pi_1} M_{\Pi_2}, \text{ where } \Pi_1, \Pi_2 \text{ are the major and minor premises of } \Pi, \\ & \text{respectively.} \\ (\mathrm{val}_0) & M_\Pi = \bot. \\ (\mathrm{val}_{>0}) & M_\Pi = \bigsqcup\{M_{\Pi_i} \mid 1 \le i \le n\}, \text{ where } (\Pi'_i)_{1\le i \le n} \text{ are the premises of } \Pi. \end{array}$$

In the case $(\mathrm{val}_{>0})$*, notice that* $\uparrow\{M_{\Pi_i} \mid 1 \le i \le n\}$*, whence its supremum is well-defined. Whenever $M_\Pi \in \mathcal{A}$, we rather call this term A_Π to stress the fact that it is an approximant.*

Intuitively, $\Pi \rhd \Gamma \vdash M : \alpha$ is in typed v-nf if no redex occurrence in M is fully typed in Π.

▶ **Lemma 21.**
(i) *For all $A \in \mathcal{A}$, there exist $\alpha \in \mathbb{T}$ and Γ such that $\Gamma \vdash A : \alpha$.*
(ii) *For all $A \in \mathcal{A}$ and $N \in \Lambda$, $\Gamma \vdash A : \alpha$ and $A \sqsubseteq_\bot N$ entail $\Gamma \vdash N : \alpha$.*

Proof. Both items follow by a straightforward induction on the structure of A. ◀

▶ **Lemma 22.** *If $\Pi \rhd \Gamma \vdash N : \alpha$ is in typed* v*-nf, then $M_\Pi \in \mathcal{A}(N)$ and $\Gamma \vdash M_\Pi : \alpha$.*

Proof. Straightforward induction on the structure of Π. ◀

▶ **Theorem 23** (Approximation Theorem)**.** *Let $M \in \Lambda$, $\alpha \in \mathbb{T}$ and Γ be an environment.*

$$\Gamma \vdash M : \alpha \iff \exists A \in \mathcal{A}(M) \,.\, \Gamma \vdash A : \alpha$$

Proof. ($\Rightarrow$) Assume $\Gamma \vdash M : \alpha$. By weighted subject reduction (Proposition 14(i)), $M \twoheadrightarrow_{\mathsf{v}} N$ for some $N \in \Lambda$ such that there exists $\Pi \rhd \Gamma \vdash N : \alpha$ in typed v-nf. Conclude by Lemma 22.

($\Leftarrow$) Assume $\Gamma \vdash A : \alpha$ for some $A \in \mathcal{A}(M)$. By definition, $M \twoheadrightarrow_{\mathsf{v}} N$ for some N satisfying $A \sqsubseteq_\bot N$. By Lemma 21(ii), $\Gamma \vdash N : \alpha$. Conclude by subject expansion (Lemma 14(ii)). ◀

▶ **Corollary 24.** *If $M, N \in \Lambda$ have the same CbV Böhm trees then $\mathcal{M} \models M = N$.*

Proof. Assume $\mathcal{A}(M) = \mathcal{A}(N)$. By applying the Approximation Theorem 23, we get $[\![M]\!] = \bigcup_{A\in\mathcal{A}(M)}[\![A]\!] = \bigcup_{A\in\mathcal{A}(N)}[\![A]\!] = [\![N]\!]$. As a consequence, we conclude $\mathcal{M} \models M = N$. ◀

3 Characterizations of Operational Properties

We now provide two characterizations of the most significant properties of the calculus, namely valuability, potential valuability and solvability. The former is logical, through the type assignment system, the latter semantic, through the Approximation Theorem.

▶ **Theorem 25** (Characterizations of valuability and potential valuability)**.** *Let $M \in \Lambda$, then:*

1. *M is valuable*	$\iff$	$\vdash M : []$	$\iff$	$\bot \in \mathcal{A}(M)$.
2. *M is potentially valuable*	$\iff$	$\exists \Gamma, \alpha \,.\, \Gamma \vdash M : \alpha$	$\iff$	$\mathcal{A}(M) \neq \emptyset$.

Proof. See [20] for the logical characterizations, and [19] for the semantic ones. ◀

To characterize solvability, we need a deeper analysis of the structure of the approximants.

▶ **Definition 26.** *The subsets $\mathcal{S},\mathcal{U} \subseteq \mathcal{A}$ are defined inductively by the grammars (for $n \geq 0$):*

$$\begin{array}{lllllllll}(\mathcal{S}) & S & ::= & H' \mid R' & \qquad & (\mathcal{U}) & U & ::= & \bot \mid \lambda x.U \\ & H' & ::= & x \mid \lambda x.S \mid xHA_1 \cdots A_n & & & & \mid & (\lambda x.U)(yHA_1 \cdots A_n) \\ & R' & ::= & (\lambda x.S)(yHA_1 \cdots A_n) & & & & & \end{array}$$

Note that $\{\mathcal{S},\mathcal{U}\}$ constitutes a partition of $\mathcal{A}$, namely $\mathcal{A} = \mathcal{S} \cup \mathcal{U}$ and $\mathcal{S} \cap \mathcal{U} = \emptyset$.

▶ **Example 27.**

(i) $x, \mathbf{I}, x\mathbf{K}\bot, \mathbf{I}(zz), \mathbf{\Delta}(zz), \mathbf{K}(y\mathbf{I}\bot), (\lambda x.(\mathbf{I}(yz)))(zy\bot) \in \mathcal{S}$.

(ii) $\bot, \lambda x_0 \ldots x_n.\bot, (\lambda x.\bot)(zz), (\lambda x.\bot)(y\mathbf{II}), (\lambda x.(\lambda y.\bot)(wz))(zw) \in \mathcal{U}$.

(iii) Finally, notice that $\mathcal{A}(\mathbf{\Omega}), \mathcal{A}(\mathbf{ZI}), \mathcal{A}(\lambda x.\mathbf{\Omega}), \mathcal{A}(\mathbf{K}^\star) \subseteq \mathcal{U}$.

We are going to show that the existence of an approximant $A \in \mathcal{A}(M)$ of shape S is enough to ensure the solvability of M. Conversely, when M is unsolvable, $\mathcal{A}(M)$ is only populated by approximants of shape U. We need a couple of technical lemmas.

▶ **Lemma 28** (Substitution Lemma)**.** *Let $M \in \Lambda$, $\mathcal{A}(M) \neq \emptyset$ and $\vec{x} = \{x_1, \ldots, x_i\} \supseteq \mathrm{FV}(M)$. Then, for all $j \geq 0$ large enough and $n_1, \ldots, n_i \geq j$, we have*

$$M[\mathbf{P}_{n_1}/x_1, \ldots, \mathbf{P}_{n_i}/x_i] \twoheadrightarrow_{\mathsf{v}} V, \textit{ for some } V \in \mathrm{Val} \cap \Lambda^o.$$

Moreover, if $x_m HA_1 \cdots A_n \in \mathcal{A}(M)$ then we can take $V = \mathbf{P}_\ell$, for $\ell = n_m - n - 1 \geq 0$.

Proof. If $A \in \mathcal{A}(M)$, then there is $N \in \Lambda$ such that $M \twoheadrightarrow_{\mathsf{v}} N$ and $A \sqsubseteq_\bot N$. By Fact 2, setting $\vartheta = [\mathbf{P}_{n_1}/x_1, \ldots, \mathbf{P}_{n_i}/x_i]$, we have $M^\vartheta \twoheadrightarrow_{\mathsf{v}} N^\vartheta \in \Lambda^o$. It suffices to check $N^\vartheta \twoheadrightarrow_{\mathsf{v}} V$.

By structural induction on A.

Case $A = x_m$ for some $m\,(1 \leq m \leq i)$. Then $N = x_m$, so $N^\vartheta = \mathbf{P}_{n_m}$ and we are done.

Case $A = \lambda y.A_0$. Then $N = \lambda y.N_0$ with $y \notin \vec{x}$ (wlog), whence $N^\vartheta = \lambda y.N_0^\vartheta \in \mathrm{Val}$.

Case $A = \bot$. Since $\bot \sqsubseteq_\bot N$ entails $N \in \mathrm{Val}$, we have either $N = x_m$ or $N = \lambda y.N_0$. Therefore, we proceed as above.

Case $A = x_m HA_1 \cdots A_n$ for $m\,(1 \leq m \leq i)$. Then $A \sqsubseteq_\bot N$ entails $N = x_m N_0 \cdots N_n$ with $H \in \mathcal{A}(N_0)$ and $A_r \in \mathcal{A}(N_r)$ for all $r\,(1 \leq r \leq n)$. Assuming $j > n$, we obtain

$$\begin{array}{rcll} N^\vartheta & = & \mathbf{P}_{n_m} N_0^\vartheta \cdots N_n^\vartheta, & \text{by definition of } \vartheta, \\ & \twoheadrightarrow_{\mathsf{v}} & \mathbf{P}_{n_m} V_0 \cdots V_n, & \text{by I.H. (induction hypothesis)}, \\ & \twoheadrightarrow_{\beta_{\mathsf{v}}} & \mathbf{P}_{n_m - n - 1}, & \text{with } n_m - n - 1 \geq 0, \text{ since } n_m \geq j > n. \end{array}$$

Case $A = (\lambda y.A_0)(xHA_1 \cdots A_n)$ with $x \in \vec{x}$ and, wlog, $y \notin \vec{x}$. From $A \sqsubseteq_\bot N$, we derive $N = (\lambda y.N_0)N_1$ where $A_0 \in \mathcal{A}(N_0)$ and $xHA_1 \cdots A_n \in \mathcal{A}(N_1)$. Easy calculations give:

$$\begin{array}{rll} N^\vartheta = & (\lambda y.N_0^\vartheta)N_1^\vartheta, & \text{since } y \notin \mathrm{dom}(\vartheta), \text{ then for some } \ell_1 \geq 0 \text{ we get:} \\ \twoheadrightarrow_{\mathsf{v}} & (\lambda y.N_0^\vartheta)\mathbf{P}_{\ell_1}, & \text{as the I.H. on } N_1 \text{ gives } N_1^\vartheta \twoheadrightarrow_{\mathsf{v}} \mathbf{P}_{\ell_1} \text{ since } xHA_1 \cdots A_n \in \mathcal{A}(N_1), \\ \to_{\mathsf{v}} & N_0^\vartheta[\mathbf{P}_{\ell_1}/y], & \text{by } (\beta_{\mathsf{v}}), \\ \twoheadrightarrow_{\mathsf{v}} & V, & \text{by applying the I.H. to } N_0 \text{ and } \vartheta \circ [\mathbf{P}_{\ell_1}/y]. \end{array}$$

◀

▶ **Proposition 29** (Context Lemma)**.** *Let $M \in \Lambda$ and $\{x_1, \ldots, x_i\} \supseteq \mathrm{FV}(M)$. If $A \in \mathcal{A}(M) \cap \mathcal{S}$ then, for all $j \geq 0$ large enough, there is $k \geq 0$ such that for all $n_1, \ldots, n_{i+k} \geq j$ we have*

$$M[\mathbf{P}_{n_1}/x_1, \ldots, \mathbf{P}_{n_i}/x_i]\mathbf{P}_{n_{i+1}} \cdots \mathbf{P}_{n_{i+k}} \twoheadrightarrow_{\mathsf{v}} \mathbf{P}_\ell, \textit{ for some } \ell \geq 0.$$

Proof. Since $A \in \mathcal{A}(M)$, there exists $N \in \Lambda$ such that $M \twoheadrightarrow_{\mathsf{v}} N$ and $A \sqsubseteq_\bot N$. Now, setting $\vartheta = [\mathbf{P}_{n_1}/x_1, \ldots, \mathbf{P}_{n_i}/x_i]$, we have $M^\vartheta \twoheadrightarrow_{\mathsf{v}} N^\vartheta$. Proceed by structural induction on $A \in \mathcal{S}$.

Case $A = x$. Take $k = 0$ and proceed as in the proof of Lemma 28.

Case $A = xHA_1 \cdots A_n$. Again, take $k = 0$ and apply Lemma 28.

Case $A = \lambda y.S$. Then $N = \lambda y.N_0$ with $y \notin \vec{x}$ and $S \in \mathcal{A}(N_0)$. By induction hypothesis, there is $k' \geq 0$ such that $n_1, \dots, n_{i+k'+1} \geq j$ entails $N_0^\vartheta[\mathbf{P}_{n_{i+1}}/y]\mathbf{P}_{n_{i+2}} \cdots \mathbf{P}_{n_{i+k'+1}} \twoheadrightarrow_{\mathsf{v}} \mathbf{P}_\ell$, for some $\ell \geq 0$. Taking $k = k' + 1$, easy calculations give $(\lambda y.N_0)^\vartheta \mathbf{P}_{n_{i+1}} \cdots \mathbf{P}_{n_{i+k}} \twoheadrightarrow_{\mathsf{v}} \mathbf{P}_\ell$.

Case $A = (\lambda y.S)(x_m H A_1 \cdots A_n)$ with $1 \leq m \leq i$ and, wlog, $y \notin \vec{x}$. From $A \sqsubseteq_\perp N$, we obtain $N = (\lambda y.N_0)N_1$ with $S \in \mathcal{A}(N_0)$, $\mathrm{FV}(N_0) \subseteq \{\vec{x}, y\}$, and $x_m H A_1 \cdots A_n \in \mathcal{A}(N_1)$. By induction hypothesis, for all j' large enough, there is k' such that for all $h_1, \dots, h_{i+k'+1} \geq j'$ we have $N_0[\mathbf{P}_{h_1}/x_1, \dots, \mathbf{P}_{h_i}/x_i, \mathbf{P}_{h_{i+1}}/y]\mathbf{P}_{h_{i+2}} \cdots \mathbf{P}_{h_{i+k'+1}} \twoheadrightarrow_{\mathsf{v}} \mathbf{P}_\ell$, for some $\ell \geq 0$. Therefore, taking $k = k' + 1$, we obtain, for all $j \geq j' + n + 1$ and $n_1, \dots, n_{i+k} \geq j$, the following:

$$
\begin{array}{rcll}
N^\vartheta \mathbf{P}_{n_{i+1}} \cdots \mathbf{P}_{n_{i+k}} & = & (\lambda y.N_0^\vartheta)N_1^\vartheta \mathbf{P}_{n_{i+1}} \cdots \mathbf{P}_{n_{i+k}}, & \text{as } y \notin \mathrm{dom}(\vartheta), \\
& \twoheadrightarrow_{\mathsf{v}} & (\lambda y.N_0^\vartheta)\mathbf{P}_{n_m - n - 1}\mathbf{P}_{n_{i+1}} \cdots \mathbf{P}_{n_{i+k}}, & \text{by Lemma 28}, \\
& \to_{\mathsf{v}} & N_0^\vartheta[\mathbf{P}_{\ell'}/y]\mathbf{P}_{n_{i+1}} \cdots \mathbf{P}_{n_{i+k}}, & \text{setting } \ell' = n_m - n - 1, \\
& \twoheadrightarrow_{\mathsf{v}} & \mathbf{P}_\ell, & \text{by I.H. since } \ell' \geq j'.
\end{array}
$$
◀

▶ **Corollary 30.** *Let $M \in \Lambda$ and $A \in \mathcal{A}(M)$. If $A \in \mathcal{S}$ then M is solvable.*

Proof. Assume $A \in \mathcal{A}(M) \cap \mathcal{S}$ and $\mathrm{FV}(M) = \{\vec{x}\}$. By Proposition 29, there are $P_1, \dots, P_k \in \Lambda^o$ such that $(\lambda \vec{x}.M)\vec{P} \twoheadrightarrow_{\mathsf{v}} \mathbf{P}_n$ for some $n \geq 0$. By applying the identity n times, we get $(\lambda \vec{x}.M)\vec{P}\,\mathbf{I}^{\sim n} \twoheadrightarrow_{\mathsf{v}} \mathbf{I}$. We conclude that M is solvable. ◀

▶ **Definition 31** (Proper type). *A type α is* trivial *if it has the following shape (for $n \geq 0$):*

$$\alpha = \sigma_1 \to \cdots \to \sigma_n \to [\,]$$

The type α is called proper *if it is not trivial.*

▶ **Example 32.**

(i) Every atom $\mathsf{a} \in \mathbb{A}$ is proper.

(ii) The following types are proper: $[\,] \to \mathsf{a}$, $[\mathsf{a}] \to \mathsf{a}$, $[[\,] \to [\,]] \to \mathsf{a}$ and $[\mathsf{a}, \mathsf{a}] \to \mathsf{a}$.

(iii) The following types are trivial: $[\,] \to [\,]$, $[\mathsf{a}] \to [\,]$, $[[\,] \to [\,]] \to [\,]$ and $[\mathsf{a}, \mathsf{a}] \to [\,]$.

▶ Remark 33. If $\alpha \in \mathbb{T}$ is proper (resp. trivial), then so is $\sigma \to \alpha$ for all $\sigma \in \mathbb{T}^!$.

We show that solvable terms admit proper types in appropriate type environments. Conversely, unsolvables are either not typable or they only admit trivial types.

▶ **Lemma 34.** *Let $M \in \Lambda$. If M is solvable then there exist an environment Γ and a proper type α such that $\Gamma \vdash M : \alpha$ is derivable.*

Proof. Assume M solvable and let $\mathrm{FV}(M) = \{x_1, \dots, x_k\}$. By definition of solvability, there exist $V_1, \dots, V_n \in \mathrm{Val}$ such that $(\lambda \vec{x}.M)\vec{V} \twoheadrightarrow_{\mathsf{v}} \mathbf{I}$. Now, for β proper, we have $\vdash \mathbf{I} : [\beta] \to \beta$. By subject expansion (Proposition 14(ii)), there is a derivation $\Pi \triangleright \vdash (\lambda \vec{x}.M)\vec{V} : [\beta] \to \beta$. If $n = k = 0$ then $M \twoheadrightarrow_{\mathsf{v}} \mathbf{I}$ and we are done taking $\Gamma = \emptyset$ and $\alpha = \beta$. Otherwise, we split into cases depending on the values of n, k. By Property 8, only the following cases are possible.

- Subcase $k = n + 1$. For some $\Gamma = x_1 : \sigma_1, \dots, x_{k-1} : \sigma_{k-1}, x_k : [\beta]$, Π must have shape:

$$
\dfrac{\dfrac{\dfrac{\Pi_0}{\Gamma \vdash M : \beta}}{\vdash \lambda \vec{x}.M : \sigma_1 \to \cdots \to \sigma_n \to [\beta] \to \beta}\,(\mathrm{lam}) \quad \dfrac{\Pi_1}{\vdash V_1 : \sigma_1} \cdots \dfrac{\Pi_n}{\vdash V_n : \sigma_n}}{\vdash (\lambda \vec{x}.M)V_1 \cdots V_n : [\beta] \to \beta}\,(\mathrm{app})
$$

We found a derivation $\Pi_0 \triangleright \Gamma \vdash M : \beta$, so we conclude because β is proper.

- Case $k \le n > 0$. For some $\Gamma = x_1 : \sigma_1, \dots, x_k : \sigma_k$, Π must have the following shape:

$$\dfrac{\dfrac{\dfrac{\Pi_0}{\Gamma \vdash M : \sigma_{k+1} \to \cdots \to \sigma_n \to [\beta] \to \beta}}{\vdash \lambda\vec{x}.M : \sigma_1 \to \cdots \to \sigma_n \to [\beta] \to \beta}\,(\mathrm{lam}) \quad \dfrac{\Pi_1}{\vdash V_1 : \sigma_1} \quad \cdots \quad \dfrac{\Pi_n}{\vdash V_n : \sigma_n}}{\vdash (\lambda\vec{x}.M)V_1 \cdots V_n : [\beta] \to \beta}\,(\mathrm{app})$$

Thus, we can take $\alpha = \sigma_{k+1} \to \cdots \to \sigma_n \to [\beta] \to \beta$, which is proper by Remark 33. ◀

▶ **Lemma 35.** *For every $A \in \mathcal{A}$, we have:*
(i) $A \in \mathcal{S} \iff \exists \Gamma, \alpha \,.\, \Gamma \vdash A : \alpha$, *with α proper.*
(ii) $A \in \mathcal{U} \iff \forall \Gamma, \alpha \,.\, \Gamma \vdash A : \alpha$ *implies that α is trivial.*

Proof. It is enough to show that ($\Rightarrow$) holds for (i) and (ii). The converse implication follows taking the contrapositive and using the facts that $\mathcal{U} = \mathcal{A} - \mathcal{S}$ and $\mathcal{S} = \mathcal{A} - \mathcal{U}$, respectively.

(i) By induction on the structure of $A \in \mathcal{S}$ (following the grammar in Definition 26).

Case $A = x$. For every $\mathsf{a} \in \mathbb{A}$, which is a proper type, we have $x : [\mathsf{a}] \vdash x : \mathsf{a}$ by (var).

Case $A = \lambda x.S$. By I.H., there exist $\Gamma, x : \sigma$ and a proper type α such that $\Gamma, x : \sigma \vdash S : \alpha$. Thus $\Gamma \vdash \lambda x.S : \sigma \to \alpha$ is derivable by (lam), where $\sigma \to \alpha$ is a proper type by Remark 33.

Case $A = xHA_1 \cdots A_n$. In this case we can assign A any type β, in the appropriate Γ. By Lemma 21(i), there are environments $\Gamma_0, \dots, \Gamma_n$ and types $\alpha_0, \dots, \alpha_n$ such that $\Gamma_0 \vdash H : \alpha_0$ and $\Gamma_i \vdash A_i : \alpha_i$ for all $i\,(1 \le i \le n)$. Setting $\Gamma = \sum_i \Gamma_i + [x : [[\alpha_0] \to \cdots \to [\alpha_n] \to \beta]]$, we get $\Gamma \vdash xHA_1 \cdots A_n : \beta$ via ($\mathrm{val}_{>0}$) and (app). We conclude by taking, e.g., $\beta = \mathsf{a} \in \mathbb{A}$.

Case $A = (\lambda y.S)(xHA_1 \cdots A_n)$. By I.H., there exist Γ_0 and a proper type α such that $\Gamma_0 \vdash S : \alpha$. Let $\Gamma_0(y) = [\alpha_1, \dots, \alpha_k]$ with $k \ge 0$, then there are environments $\Gamma_1, \dots, \Gamma_k$ such that $\Gamma_i \vdash xHA_1 \cdots A_n : \alpha_i$ for all $i\,(1 \le i \le k)$, as we have seen above that such term can be assigned any type. Taking $\Gamma = \sum_{i=0}^{n} \Gamma_i$, we conclude $\Gamma \vdash A : \alpha$ where α is proper.

(ii) By induction on the structure of $A \in \mathcal{U}$ (following the grammar in Definition 26).

Case $A = \bot$. The only applicable rule is (val_0), namely $\vdash \bot : []$.

Case $A = \lambda x.U$. Assume that $\Gamma \vdash \lambda x.U : \sigma \to \alpha$ holds, then also $\Gamma, x : \sigma \vdash U : \alpha$ is derivable. By I.H. the type α is trivial, therefore $\sigma \to \alpha$ is also trivial by Remark 33.

Case $A = (\lambda y.U)(xHA_1 \cdots A_n)$. Assume that $\Gamma \vdash A : \alpha$ holds, then there exists a decomposition $\Gamma = \Gamma_0 + \Gamma_1$ and a $\sigma \in \mathbb{T}^!$ such that $\Gamma_0, y : \sigma \vdash U : \alpha$ and $\Gamma_1 \vdash xHA_1 \cdots A_n : \sigma$. By applying the I.H. on $\Gamma_0, y : \sigma \vdash U : \alpha$, we conclude that α is trivial. ◀

▶ **Theorem 36** (Characterizations of solvability)**.** *For $M \in \Lambda$, the following are equivalent:*
1. *M is solvable.*
2. *There exists a proper type α such that $\Gamma \vdash M : \alpha$, for some environment Γ.*
3. *There exists an approximant $A \in \mathcal{A}(M) \cap \mathcal{S}$.*

Proof. ($1 \Rightarrow 2$) By Lemma 34.

($2 \Rightarrow 3$) By the Approximation Theorem, there exists $A \in \mathcal{A}(M)$ such that $\Gamma \vdash M : \alpha$. By Lemma 35(i), we derive $A \in \mathcal{S}$.

($3 \Rightarrow 1$) By Corollary 30. ◀

▶ **Corollary 37.** *A λ-term M is unsolvable exactly when $\mathcal{A}(M) \subseteq \mathcal{U}$, equivalently, whenever $\Gamma \vdash M : \alpha$ entails that α is a trivial type.*

▶ **Corollary 38.** *The model $\mathcal{M}$ is not sensible, but semi-sensible.*

Proof. The model is not sensible as $[\![\mathbf{\Omega}]\!] = \emptyset$ and $[\![\lambda x.\mathbf{\Omega}]\!] = \{[]\}$, entail $\mathcal{M} \not\models \mathbf{\Omega} = \lambda x.\mathbf{\Omega}$. If M is solvable and N is unsolvable, by Theorem 36 there exist an environment Γ and a type α proper such that $(\Gamma, \alpha) \in [\![M]\!] - [\![N]\!]$, therefore the model is semi-sensible. ◀

$$\frac{}{\bot \in \mathsf{IM}(\emptyset;\, [])}\ (\mathsf{bot}^!) \qquad \frac{A_i \in \mathsf{IT}(\Gamma_i;\, \alpha_i) \quad \uparrow\{A_i\}_{i\in I} \quad A = \bigsqcup_{i\in I} A_i}{A \in \mathsf{IM}(\Sigma_{i\in I}\Gamma_i;\, [\alpha_i]_{i\in I})}\ (\mathsf{sup}^!)$$

$$\frac{}{\bot \in \mathsf{IT}(\emptyset;\, [])}\ (\mathsf{bot}) \qquad \frac{A \in \mathsf{IT}(\Gamma, x : \sigma;\, \alpha)}{\lambda x.A \in \mathsf{IT}(\Gamma;\, \sigma \to \alpha)}\ (\mathsf{abs}) \qquad \frac{}{x \in \mathsf{IT}(x : [\alpha];\, \alpha)}\ (\mathsf{head}_0)$$

$$\frac{A_j \in \mathsf{IM}(\Gamma_j;\, \sigma_j) \quad 0 \le j \le n \quad A_0 \in \mathcal{H}}{xA_0 \cdots A_n \in \mathsf{IT}(\sum_{j=0}^{n} \Gamma_j + x : [\sigma_0 \to \cdots \to \sigma_n \to \alpha];\, \alpha)}\ (\mathsf{head}_{>0})$$

$$\frac{A_j \in \mathsf{IM}(\Gamma_j;\, \sum_{i=0}^{m} \tau_j^i) \quad 0 \le j \le n \quad A_0 \in \mathcal{H} \quad A \in \mathsf{IT}(\Gamma_{n+1}, x : [\alpha_i]_{0\le i\le m};\, \alpha)}{(\lambda x.A)(yA_0 \cdots A_n) \in \mathsf{IT}(\sum_{j=0}^{n+1} \Gamma_j + y : [\tau_0^i \to \cdots \to \tau_n^i \to \alpha_i]_{0\le i\le m};\, \alpha)}\ (\mathsf{redlike})$$

Figure 2 The inhabitation algorithm for system $\mathcal{M}$. In $(\mathsf{redlike})$, we assume $x \notin \mathrm{FV}(yA_0 \cdots A_n)$.

4 Decidability of the Inhabitation Problem

The inhabitation problem for system $\mathcal{M}$ requires to determine for every environment Γ and type α whether there is a λ-term M satisfying $\Gamma \vdash M : \alpha$. To show that this problem is decidable we describe an algorithm that takes (Γ, α) as input and returns as output the set of all approximants A satisfying $\Gamma \vdash A : \alpha$ as well as the following minimality condition.

▶ **Definition 39.** *Let Γ be an environment and $\xi \in \mathbb{T} \cup \mathbb{T}^!$. An $A \in \mathcal{A}$* is minimal for (Γ, ξ) *if $\Gamma \vdash A : \xi$ and, for all $A' \in \mathcal{A}$ compatible with A (i.e. $\uparrow\{A, A'\}$), $\Gamma \vdash A' : \xi$ entails $A \sqsubseteq_\bot A'$.*

Finding the minimal approximants inhabiting (Γ, α) is enough for solving the original inhabitation problem because $\Gamma \vdash M : \alpha$ holds exactly when there is an $A \in \mathcal{A}(M)$ minimal for (Γ, α). Following [9, 8], we present the inhabitation algorithm as a deductive system.

▶ **Definition 40.**

(i) *Let Γ be an environment and $\alpha \in \mathbb{T}$. The* inhabitation algorithm $\mathsf{IT}(\Gamma;\, \alpha)$ *for $\mathcal{M}$ is given in Figure 2, via an auxiliary predicate $\mathsf{IM}(\Gamma;\, \sigma)$, for $\sigma \in \mathbb{T}^!$. Note that the condition $A_0 \in \mathcal{H}$ occurring as a premise of the rules $(\mathsf{head}_{>0})$ and $(\mathsf{redlike})$ is decidable since $\mathcal{H}$ is generated by a context-free grammar (Definition 17*(ii)*).*

(ii) *A* run *of the algorithm is a deduction tree built bottom-up by applying the rules in Figure 2 in such a way that every node is an instance of a rule (as in Example 41). We say that* a run of the algorithm terminates *if such a tree is finite. The* algorithm terminates *if it needs to execute a finite number of different terminating runs.*

It is easy to check that $A \in \mathsf{IT}(\Gamma;\, \alpha)$ (resp. $A \in \mathsf{IM}(\Gamma;\, \sigma)$) implies $\mathrm{FV}(A) \subseteq \mathrm{dom}(\Gamma)$. We are going to prove that the inhabitation algorithm is terminating, sound and complete. Completeness is achieved by exploiting the non-determinism of the algorithm: indeed, when $\alpha = \sigma \to \beta$, the rules (abs), $(\mathsf{redlike})$ and $(\mathsf{head}_{\_})$ might be applicable and in $(\mathsf{redlike})$ and $(\mathsf{head}_{>0})$, the environment Γ can be decomposed in countably many different ways (taking many $\Gamma_i = \emptyset$). By collecting all possible runs, we recover all minimal approximants for (Γ, α).

▶ **Example 41.** The following are examples of possible runs of the algorithm on $\mathsf{IT}(\Gamma;\, \alpha)$.

(i) Let $\Gamma = y : [[] \to \mathsf{a}]$ and $\alpha = \mathsf{a}$. There are two runs:

$$\frac{\dfrac{}{\bot \in \mathsf{IM}(\emptyset;\, [])}\,(\mathsf{bot}^!) \quad \bot \in \mathcal{H} \quad \dfrac{}{x \in \mathsf{IT}(x : [\mathsf{a}];\, \mathsf{a})}\,(\mathsf{head}_0)}{(\lambda x.x)(y\bot) \in \mathsf{IT}(y : [[] \to \mathsf{a}];\, \mathsf{a})}\ (\mathsf{redlike}) \qquad \frac{\dfrac{}{\bot \in \mathsf{IM}(\emptyset;\, [])}\,(\mathsf{bot}^!) \quad \bot \in \mathcal{H}}{y\bot \in \mathsf{IT}(y : [[] \to \mathsf{a}];\, \mathsf{a})}\ (\mathsf{head}_{>0})$$

(ii) Let $\Gamma = y : [[] \to []]$ and $\alpha = [\mathsf{a}] \to \mathsf{a}$. The only possible run is $\mathsf{redlike}(\mathsf{abs}(\mathsf{head}_0), \mathsf{bot}^!)$ which constructs the approximant $(\lambda x.\mathbf{I})(y\bot)$.

(iii) Let $\Gamma = \emptyset$ and $\alpha = [[\mathsf{a}] \to \mathsf{a}, [\mathsf{a}] \to \mathsf{a}] \to [\mathsf{a}] \to \mathsf{a}$. Also in this case, the only possible run is $\mathsf{abs}(\mathsf{abs}(\mathsf{head}_{>0}(\mathsf{sup}^!(\mathsf{head}_{>0}(\mathsf{sup}^!(\mathsf{head}_0))))))$, which constructs $\lambda xy.x(xy)$.

(iv) Let $\Gamma = \emptyset$ and $\alpha = [[\mathsf{a}] \to \mathsf{a}] \to [\mathsf{a}] \to \mathsf{a}$. The run $\mathsf{abs}(\mathsf{head}_0)$ constructs $\lambda x.x$, while the run $\mathsf{abs}(\mathsf{abs}(\mathsf{head}_{>0}(\mathsf{sup}^!(\mathsf{head}_0))))$ constructs $\lambda xy.xy$.

(v) Let $\Gamma = x : [[] \to [] \to \mathsf{a}]$ and $\alpha = \mathsf{a}$. There are two possible runs: $\mathsf{head}_{>0}(\mathsf{bot}^!, \mathsf{bot}^!)$, building $x\bot\bot$, and $\mathsf{redlike}(\mathsf{bot}^!, \mathsf{head}_0)$, building $(\lambda z.z)(x\bot\bot)$.

▶ **Definition 42.** *To show that the inhabitation algorithm terminates we define two* measures, *$\#(\cdot)$ on types, and $(\cdot)^\bullet$ on multiset types and type environments, as follows (for $\mathsf{a} \in \mathbb{A}, n \geq 0$):*

$$\begin{array}{lllllll} \#\mathsf{a} & = & \#[] & = & 1, & \#(\sigma \to \alpha) & = & \sigma^\bullet + \#\alpha + 3, \\ [\alpha_1, \dots, \alpha_n]^\bullet & = & \sum_{i=1}^n \#\alpha_i, & & & \Gamma^\bullet & = & \sum_{x \in \mathrm{dom}(\Gamma)} \Gamma(x)^\bullet. \end{array}$$

Note that $\#\alpha \geq 1$, while $[]^\bullet = 0$. If $\sigma = \sigma_1 + \sigma_2$ then $\sigma^\bullet = \sigma_1^\bullet + \sigma_2^\bullet$, thus $\Gamma = \sum_{i \in I} \Gamma_i$ entails $\Gamma^\bullet = \sum_{i\in I} \Gamma_i^\bullet$. The measure $\#(\cdot)$ is extended to judgements $\mathsf{IT}(-; -)$ and $\mathsf{IM}(-; -)$ by

$$\#(\mathsf{IT}(\Gamma; \alpha)) = \Gamma^\bullet + \#\alpha, \qquad \#(\mathsf{IM}(\Gamma; \sigma)) = \Gamma^\bullet + \sigma^\bullet + 1.$$

Given $M \in \Lambda_\bot$, we define inductively the size of its syntax-tree, *written* $\mathsf{tsize}(M)$, *by:*

$$\mathsf{tsize}(\bot) = \mathsf{tsize}(x) = 0, \quad \mathsf{tsize}(\lambda x.P) = \mathsf{tsize}(P) + 1, \quad \mathsf{tsize}(PQ) = \mathsf{tsize}(P) + \mathsf{tsize}(Q) + 1.$$

▶ **Example 43.**

(i) We have $\#([] \to []) = \#([] \to \mathsf{a}) = 4$, so $(x : [[] \to [], [] \to \mathsf{a}, \mathsf{a}])^\bullet = 9$.

(ii) Since $\#[[\mathsf{a}] \to \mathsf{a}] = 5$, we get $\#\mathsf{IT}(x : [[\mathsf{a}] \to \mathsf{a}]; \mathsf{a}) = 6$, while $\#\mathsf{IM}(x : [[\mathsf{a}] \to \mathsf{a}]; [\mathsf{a}]) = 7$.

(iii) $\mathsf{tsize}((\lambda x.\bot)(x\bot)) = 3$, $\mathsf{tsize}(\mathbf{P}_n) = n + 1$ and $\mathsf{tsize}(x\bot^{\sim n}) = n$, for all $n \geq 0$.

▶ **Lemma 44.** *Every run of the inhabitation algorithm terminates.*

Proof. We need to show that every run is a finite tree. Since we are considering finite multisets and all indices range over $\mathbb{N}$, the premises of each rule in Figure 2 are finitely many (i.e., a run is a finitely branching tree), whence it is enough to show that there is no infinite path (by König's Lemma). This follows from the fact that the measure # calculated on each premise of a rule, is strictly smaller than the measure associated with its conclusion. We proceed by cases on the rules applied, the cases (bot), $(\mathsf{bot}^!)$, and (head_0) being vacuous.

Cases (abs) and $(\mathsf{sup}^!)$ follow straightforwardly from Definition 42.

Case $(\mathsf{head}_{>0})$ with premises $\mathsf{IM}(\Gamma_j; \sigma_j)$, for all $j\,(0 \leq j \leq n)$, and as a conclusion $\mathsf{IT}(\Gamma + x : [\sigma_0 \to \cdots \to \sigma_n \to \alpha]; \alpha)$ for $\Gamma = \sum_{j=0}^n \Gamma_j$. The measure # applied to the j-th premise gives $\Gamma_j^\bullet + \sigma_j^\bullet + 1$; on the conclusion, it gives $\Gamma^\bullet + \sigma_0^\bullet + \cdots + \sigma_n^\bullet + 2(\#\alpha) + 3(n+1)$. In the worse case, namely $n = j = 0$ and $\#\alpha = 1$, we still get $\Gamma_0^\bullet + \sigma_0^\bullet + 1 < \Gamma_0^\bullet + \sigma_0^\bullet + 5$.

Case $(\mathsf{redlike})$ with premises $\mathsf{IM}(\Gamma_j; \sum_{i=0}^m \tau_{ij})$, for $0 \leq j \leq n$, and $\mathsf{IT}(\Gamma_{n+1}, x : [\alpha_i]_{0 \leq i \leq m}; \alpha)$, and conclusion $\mathsf{IT}(\Gamma + y : [\tau_{i0} \to \cdots \to \tau_{in} \to \alpha_i]_{0 \leq i \leq m}; \alpha)$ for $\Gamma = \sum_{j=0}^n \Gamma_j + \Gamma_{n+1}$. For the measure applied to the conclusion, easy calculations give the following number K:

$$\begin{array}{lll} K & = & \Gamma^\bullet + 3(n+1)(m+1) + \sum_{i=0}^m (\sum_{j=0}^n \tau_{ij}^\bullet + \#\alpha_i) + \#\alpha \\ & = & \Gamma_{n+1}^\bullet + 3(n+1)(m+1) + \sum_{j=0}^n (\Gamma_j^\bullet + \sum_{i=0}^m (\tau_{ij}^\bullet + \#\alpha_i)) + \#\alpha \end{array}$$

For the j-th premise we can easily check $\Gamma_j^\bullet + \sum_{i=0}^m \tau_{ij}^\bullet + 1 < K$. For the remaining one, we get $\Gamma_{n+1}^\bullet + \sum_{i=0}^m \#\alpha_i + \#\alpha$. In the worst case, i.e. $n = m = 0$, $\Gamma^\bullet = \Gamma_0^\bullet + \Gamma_1^\bullet = \Gamma_1^\bullet$ and $\tau_{00}^\bullet = 0$, we obtain $\Gamma^\bullet + \#\alpha_0 + \#\alpha < \Gamma^\bullet + \#\alpha_0 + \#\alpha + 3$. This concludes the proof. ◀

We show that the size of the approximants generated by $\mathsf{IT}(\Gamma;\,\alpha)$ is bounded by $\Gamma^\bullet + \#\alpha$. In fact, the coefficient 3 in the definition of $\#(\sigma \to \alpha)$ has been chosen to absorb the size of the "$\lambda x.$" and of the outer application in redex-like approximants as $(\lambda x.A)(yA_0 \cdots A_n)$.

▶ **Lemma 45.** *For a type environment* Γ*,* $\alpha \in \mathbb{T}$*,* $\sigma \in \mathbb{T}^!$*, we have:*

(i) $A \in \mathsf{IT}(\Gamma;\,\alpha)$ *entails* $\mathsf{tsize}(A) \leq \#\mathsf{IT}(\Gamma;\,\alpha)$*.*

(ii) $A \in \mathsf{IM}(\Gamma;\,\sigma)$ *entails* $\mathsf{tsize}(A) < \#\mathsf{IM}(\Gamma;\,\sigma)$*.*

Proof. We prove (i) and (ii) by induction on a run of $A \in \mathsf{IT}(\Gamma;\,\alpha)$ (resp. $A \in \mathsf{IM}(\Gamma;\,\sigma)$).

Cases (bot), ($\mathsf{bot}^!$) and (head_0). Trivial, since $\mathsf{tsize}(\bot) = \mathsf{tsize}(x) = 0$ and $\mathsf{IM}(\Gamma;\,\alpha) \geq 1$.

Case ($\mathsf{sup}^!$) follows from I.H., because $A = \bigsqcup_{i\in I} A_i$ implies $\mathsf{tsize}(A) \leq \sum_{i\in I} \mathsf{tsize}(A_i)$.

Case (abs) with $\alpha = \sigma \to \beta$. By induction hypothesis, we get $\mathsf{tsize}(A) \leq \Gamma^\bullet + \sigma^\bullet + \#\beta$, therefore we obtain $\mathsf{tsize}(\lambda x.A) = \mathsf{tsize}(A) + 1 \leq \Gamma^\bullet + \sigma^\bullet + \#\beta + 3 = \#\mathsf{IT}(\Gamma;\,\sigma \to \beta)$.

Case ($\mathsf{head}_{>0}$) with $\Gamma = \sum_{j=0}^{n} \Gamma_j + x : [\sigma_0 \to \cdots \to \sigma_n \to \alpha]$. By IH, $\mathsf{tsize}(A_j) \leq \Gamma_j^\bullet + \sigma_j^\bullet$. So, $\mathsf{tsize}(xA_0 \cdots A_n) = \sum_{j=0}^{n} \mathsf{tsize}(A_j) + n + 1 \leq \sum_{j=0}^{n} \Gamma_j^\bullet + \sigma_j^\bullet + 2\#\alpha + 3(n+1) = \#\mathsf{IT}(\Gamma;\,\alpha)$.

Case ($\mathsf{redlike}$) with $\Gamma = \sum_{j=0}^{n+1} \Gamma_j + y : [\tau_{i0} \to \cdots \to \tau_{in} \to \alpha_i]_{0\leq i\leq m}$. By I.H., we have $\mathsf{tsize}(A_j) \leq \Gamma_j^\bullet + \sum_{i=0}^{m} \tau_{ij}^\bullet$ for all j ($0 \leq j \leq n$), and $\mathsf{tsize}(A) \leq \Gamma_{n+1}^\bullet + \sum_{i=0}^{m} \#\alpha_i + \#\alpha$. Thus, $\mathsf{tsize}(yA_0 \cdots A_n) = \sum_{j=0}^{n} \mathsf{tsize}(A_j) + n + 1 \leq \sum_{j=0}^{n} \Gamma_j^\bullet + \sum_{i=0}^{m} (\tau_{i0}^\bullet + \cdots + \tau_{in}^\bullet) + n + 1$ and:

$$\begin{array}{rl}
\mathsf{tsize}((\lambda x.A)(yA_0 \cdots A_n)) = & \mathsf{tsize}(\lambda x.A) + \mathsf{tsize}(yA_0 \cdots A_n) + 1 \\
\leq & \Gamma_{n+1}^\bullet + \sum_{i=0}^{m} \#\alpha_i + \#\alpha + \mathsf{tsize}(yA_0 \cdots A_n) + 2 \\
\leq & \sum_{j=0}^{n+1} \Gamma_j^\bullet + \sum_{i=0}^{m} (\tau_{i0}^\bullet + \cdots + \tau_{in}^\bullet + \#\alpha_i) + \#\alpha + n + 3 \\
\leq & \sum_{j=0}^{n+1} \Gamma_j^\bullet + \sum_{i=0}^{m} (\sum_{j=0}^{n} \tau_{ij}^\bullet + \#\alpha_i) + \#\alpha + 3(n+1)(m+1)
\end{array}$$

where the last inequation holds since $n + 3 \leq 3(n+1)(m+1)$ for all $n, m \geq 0$. ◀

▶ **Theorem 46** (Termination)**.** *The inhabitation algorithm terminates.*

Proof. Fix an input (Γ, α). By Lemma 44, every run $A \in \mathsf{IT}(\Gamma;\,\alpha)$ terminates. The set $\{A \in \mathcal{A} \mid \mathrm{FV}(A) \subseteq \mathrm{dom}(\Gamma) \wedge \mathsf{tsize}(A) \leq \Gamma^\bullet + \#\alpha\}$ is finite, because one cannot add variables or $\bot$ without adding applications. By Lemma 45, we get that the number of runs is finite. ◀

To better understand the inhabitation algorithm it is convenient to provide an effective way of constructing minimal approximants. We have seen in Definition 20(iii) that we are able to associate an approximant A_Π with every derivation Π in typed v-normal form. This last condition is always satisfied by derivations $\Pi \rhd \Gamma \vdash A : \alpha$ for $A \in \mathcal{A}$ because approximants do not contain any occurrence of a v-redex. We now show that the approximants A_Π so constructed are minimal for (Γ, α) and that all such minimal approximants arise in this way.

▶ **Lemma 47.** *Let* Γ *be a type environment,* $\alpha \in \mathbb{T}$ *and* $A \in \mathcal{A}$*. The following are equivalent:*

1. $A \in \mathsf{IT}(\Gamma;\,\alpha)$*.*
2. $A = A_\Pi$ *for some derivation* $\Pi \rhd \Gamma \vdash A : \alpha$*.*
3. A *is minimal for* (Γ, α)*.*

Proof. To perform the induction properly, we prove simultaneously the analogous statement on $\sigma \in \mathbb{T}^!$: $A \in \mathsf{IM}(\Gamma;\,\sigma) \iff A = A_\Pi$ for some $\Pi \rhd \Gamma \vdash A : \sigma \iff A$ is minimal for (Γ, σ).

($1 \Rightarrow 2$) By induction on a run of $A \in \mathsf{IT}(\Gamma;\,\alpha)$ (resp. $A \in \mathsf{IM}(\Gamma;\,\sigma)$).

Cases (bot), ($\mathsf{bot}^!$) and (head_0) are trivial.

Cases ($\mathsf{sup}^!$) and (abs). Easy. Use the I.H. and apply ($\mathrm{val}_{>0}$) and (lam), respectively.

Case $(\mathsf{head}_{>0})$ with $\Gamma = \sum_{j=0}^{n} \Gamma_j + \Gamma'$ where $\Gamma' = x : [\sigma_0 \to \cdots \to \sigma_n \to \alpha]$ and $A = xA_0 \cdots A_n$. Let $\Pi' \ \triangleright \Gamma_{n+1} \vdash x : \sigma_0 \to \cdots \to \sigma_n \to \alpha$ with $A_{\Pi'} = x$. By I.H., for every $j (0 \leq j \leq n)$, there is a derivation $\Pi_j \ \triangleright \Gamma_j \vdash A_j : \sigma_j$ such that $A_j = A_{\Pi_j}$. For Π, take:

$$\frac{\Pi' \ \triangleright \Gamma' \vdash x : \sigma_0 \to \cdots \to \sigma_n \to \alpha \quad \Pi_j \ \triangleright \Gamma_j \vdash A_j : \sigma_j \quad 0 \leq j \leq n}{\Gamma \vdash xA_0 \cdots A_n : \alpha}\ (\text{app})$$

and conclude because $A_\Pi = A_{\Pi'} A_{\Pi_0} \cdots A_{\Pi_n} = xA_0 \cdots A_n$.

Case $(\mathsf{redlike})$ with $\Gamma = \sum_{j=0}^{n+1} \Gamma_j + \Gamma'$, where $\Gamma' = y : [\tau_0^i \to \cdots \to \tau_n^i \to \alpha_i]_{0 \leq i \leq m}$, and $A = (\lambda x.A')(yA_0 \cdots A_n)$. By I.H., there exists $\Pi_{n+1} \ \triangleright \Gamma_{n+1}, x : [\alpha_i]_{0 \leq i \leq m} \vdash A' : \alpha$ with $A' = A_{\Pi_{n+1}}$. Moreover, for each $0 \leq j \leq n$, there is $\Pi_j \ \triangleright \Gamma_j \vdash A_j : \sum_{i=0}^{m} \tau_j^i$ with $A_j = A_{\Pi_j}$. This holds exactly when there exists a decomposition $\Gamma_j = \sum_{i=0}^{m} \Gamma_j^i$ and $\Pi_j^i \ \triangleright \Gamma_j^i \vdash A_j : \tau_j^i$ satisfying $A_j = \bigsqcup_{i=0}^{m} A_{\Pi_j^i}$, although individually $A_j \neq A_{\Pi_j^i}$ might hold. Construct Π as:

$$\frac{\dfrac{\begin{array}{c}\Pi_{n+1} \triangleright \\ \Gamma_{n+1}, x : [\alpha_i]_{0 \leq i \leq m} \vdash A' : \alpha\end{array}}{\Gamma_{n+1} \vdash \lambda x.A' : [\alpha_i]_{0 \leq i \leq m} \to \alpha} \quad \dfrac{\dfrac{\begin{array}{c} y : [\tau_0^i \to \cdots \to \tau_n^i \to \alpha_i] \\ \vdash y : \tau_0^i \to \cdots \to \tau_n^i \to \alpha_i\end{array} \quad \begin{array}{c}\Pi_0^i \triangleright \\ \Gamma_0^i \vdash A_0 : \tau_0^i\end{array} \ \cdots \ \begin{array}{c}\Pi_n^i \triangleright \\ \Gamma_n^i \vdash A_n : \tau_n^i\end{array}}{\sum_{j=0}^{n} \Gamma_j^i + y : [\tau_0^i \to \cdots \to \tau_n^i \to \alpha_i] \vdash yA_0 \cdots A_n : \alpha_i}\ \forall i}{\sum_{j=0}^{n} \Gamma_j + \Gamma' \vdash yA_0 \cdots A_n : [\alpha_i]_{0 \leq i \leq m}}}{\sum_{j=0}^{n+1} \Gamma_j + \Gamma' \vdash (\lambda x.A')(yA_0 \cdots A_n) : \alpha}$$

It is now easy to check that $(\lambda x.A')(yA_0 \cdots A_n) = A_\Pi$, for the derivation Π above.

$(2 \Rightarrow 3)$ By straightforward induction on $\Pi \ \triangleright \Gamma \vdash A : \alpha$. In the case $(\text{val}_{>0})$ with premises $(\Pi_i)_{i \in I}$, use the fact that $A_\Pi = \bigsqcup_{i \in I} A_{\Pi_i}$ is defined as the least upper bound.

$(3 \Rightarrow 1)$ By induction on a derivation $\Pi \ \triangleright \Gamma \vdash A : \alpha$, where A is minimal for (Γ, α). The only non-trivial case to handle is (app). We split into subcases depending on the shape of A.

Subcase $A = xA_0 \cdots A_n$ with $A_0 \in \mathcal{H}$. Then there is a decomposition $\Gamma = \sum_{j=0}^{n} \Gamma_j + x : [\sigma_0 \to \cdots \to \sigma_n \to \alpha]$ such that Π has subderivations $\Pi_j \ \triangleright \Gamma_j \vdash A_j : \sigma_j$ with A_j minimal for (Γ_j, σ_j). By I.H., $A_j \in \mathsf{IM}(\Gamma_j; \sigma_j)$ from which $xA_0 \cdots A_n \in \mathsf{IT}(\Gamma; \alpha)$ follows by $(\mathsf{head}_{>0})$.

Subcase $A = (\lambda x.A')(yHA_1 \cdots A_n)$. Then, the derivation $\Pi \ \triangleright \Gamma \vdash A : \alpha$ must have the shape above (see proof of $(1 \Rightarrow 2)$, case $(\mathsf{redlike})$) for some decomposition $\Gamma = \sum_{j=0}^{n+1} \Gamma_j + \Gamma'$, where $\Gamma' = y : [\tau_0^i \to \cdots \to \tau_n^i \to \alpha_i]_{0 \leq i \leq m}$ and setting $A_0 = H \in \mathcal{H}$. Since A is minimal for (Γ, α) and $\Gamma_j^i \vdash A_j : \tau_j^i$ for every j $(0 \leq j \leq n)$, we must have A_j minimal for $(\Gamma_j, \sum_{i=0}^{m} \tau_j^i)$ and A' minimal for $((\Gamma_{n+1}, x : [\alpha_i]_{0 \leq i \leq m}), \alpha)$. By I.H., we obtain $A_j \in \mathsf{IM}(\Gamma_j; \sum_{i=0}^{m} \tau_j^i)$ and $A' \in \mathsf{IT}((\Gamma_{n+1}, x : [\alpha_i]_{0 \leq i \leq m}); \alpha)$. As $A_0 \in \mathcal{H}$, we get $A \in \mathsf{IT}(\Gamma; \alpha)$ by applying $(\mathsf{redlike})$. ◀

▶ **Theorem 48** (Soundness and Completeness).

(i) *If $A \in \mathsf{IT}(\Gamma; \alpha)$ then, for all $M \in \Lambda$ satisfying $A \sqsubseteq_\bot M$, we have $\Gamma \vdash M : \alpha$.*

(ii) *If $\Gamma \vdash M : \alpha$ then there exists $A \in \mathsf{IT}(\Gamma; \alpha)$ such that $A \in \mathcal{A}(M)$.*

Proof. (i) By Lemma 47, we have $\Gamma \vdash A : \alpha$. Since $A \sqsubseteq_\bot M$, we conclude by Lemma 21(ii).

(ii) By the Approximation Theorem, there exists $A' \in \mathcal{A}(M)$ satisfying $\Gamma \vdash A' : \alpha$. Then, there is an approximant $A \uparrow A'$ which is minimal for (Γ, α). By Lemma 47, we obtain $A \in \mathsf{IT}(\Gamma; \alpha)$ and since $\mathcal{A}(M)$ is downward closed (by definition) we conclude $A \in \mathcal{A}(M)$. ◀

Conclusions

In this paper we have shown that the model $\mathcal{M}$ allows to characterize solvability semantically, but we believe that Theorem 36 extends to all relational models defined in [20] having a non-empty set of atoms, and whose type equivalence preserves the non-triviality of the types. The fact that $\mathcal{M}$ constitutes a model of CbV λ-calculus has been shown in [20] by exploiting the environmental definition *à la* Hindley-Longo (namely, Definition 10.0.1 in [26]). In future works, we plan to analyze the categorical construction behind this class of models as they do not seem to be an instance of any categorical definition proposed so far.

References

1 Samson Abramsky. Domain theory in logical form. *Ann. Pure Appl. Log.*, 51(1-2):1–77, 1991. `doi:10.1016/0168-0072(91)90065-T`.

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

3 Beniamino Accattoli and Giulio Guerrieri. Types of fireballs. In Sukyoung Ryu, editor, *Programming Languages and Systems – 16th Asian Symposium, APLAS 2018, Wellington, New Zealand, December 2-6, 2018, Proceedings*, volume 11275 of *Lecture Notes in Computer Science*, pages 45–66. Springer, 2018. `doi:10.1007/978-3-030-02768-1_3`.

4 Hendrik Pieter Barendregt, Wil Dekkers, and Richard Statman. *Lambda Calculus with Types*. Perspectives in logic. Cambridge University Press, 2013. URL: `http://www.cambridge.org/de/academic/subjects/mathematics/logic-categories-and-sets/lambda-calculus-types`.

5 Henk P. Barendregt. *The lambda-calculus, its syntax and semantics*. Number 103 in Studies in Logic and the Foundations of Mathematics. North-Holland, second edition, 1984.

6 O. Bastonero, Alberto Pravato, and Simona Ronchi Della Rocca. Structures for lazy semantics. In David Gries and Willem P. de Roever, editors, *Programming Concepts and Methods, IFIP TC2/WG2.2,2.3 International Conference on Programming Concepts and Methods (PROCOMET '98) 8-12 June 1998, Shelter Island, New York, USA*, volume 125 of *IFIP Conference Proceedings*, pages 30–48. Chapman & Hall, 1998.

7 Flavien Breuvart, Giulio Manzonetto, and Domenico Ruoppolo. Relational graph models at work. *Log. Methods Comput. Sci.*, 14(3), 2018. `doi:10.23638/LMCS-14(3:2)2018`.

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

9 Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. Inhabitation for non-idempotent intersection types. *Log. Methods Comput. Sci.*, 14(3), 2018. `doi:10.23638/LMCS-14(3:7)2018`.

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

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

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

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

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

15 Jean-Yves Girard. Normal functors, power series and λ-calculus. *Ann. Pure Appl. Log.*, 37(2):129–177, 1988. doi:10.1016/0168-0072(88)90025-5.

16 Giulio Guerrieri. Personal communication, 2017.

17 Giulio Guerrieri, Luca Paolini, and Simona Ronchi Della Rocca. Standardization and conservativity of a refined call-by-value lambda-calculus. *Log. Methods Comput. Sci.*, 13(4), 2017. doi:10.23638/LMCS-13(4:29)2017.

18 Furio Honsell and Marina Lenisa. Some results on the full abstraction problem for restricted lambda calculi. In Andrzej M. Borzyszkowski and Stefan Sokolowski, editors, *Mathematical Foundations of Computer Science 1993, 18th International Symposium, MFCS'93, Gdansk, Poland, August 30 – September 3, 1993, Proceedings*, volume 711 of *Lecture Notes in Computer Science*, pages 84–104. Springer, 1993. doi:10.1007/3-540-57182-5_6.

19 Emma Kerinec, Giulio Manzonetto, and Michele Pagani. Revisiting call-by-value Böhm trees in light of their Taylor expansion. *Log. Methods Comput. Sci.*, 16(3), 2020. URL: https://lmcs.episciences.org/6638.

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

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

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

23 Alberto Pravato, Simona Ronchi Della Rocca, and Luca Roversi. The call by value λ-calculus: a semantic investigation. *Mathematical Structures in Computer Science*, 9(5):617–650, 1999.

24 Laurent Regnier. Une équivalence sur les lambda-termes. *Theor. Comput. Sci.*, 126(2):281–292, 1994. doi:10.1016/0304-3975(94)90012-4.

25 Simona Ronchi Della Rocca. Intersection types and denotational semantics: An extended abstract (invited paper). In Silvia Ghilezan, Herman Geuvers, and Jelena Ivetic, editors, *22nd International Conference on Types for Proofs and Programs, TYPES 2016, May 23-26, 2016, Novi Sad, Serbia*, volume 97 of *LIPIcs*, pages 2:1–2:7. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.TYPES.2016.2.

26 Simona Ronchi Della Rocca and Luca Paolini. *The Parametric λ-Calculus: a Metamodel for Computation.* EATCS Series. Springer, Berlin, 2004.

27 Pawel Urzyczyn. The emptiness problem for intersection types. *J. Symb. Log.*, 64(3):1195–1215, 1999. doi:10.2307/2586625.

28 Pawel Urzyczyn. Personal communication, 2014.

29 Steffen van Bakel. Complete restrictions of the intersection type discipline. *Theor. Comput. Sci.*, 102(1):135–163, 1992. doi:10.1016/0304-3975(92)90297-S.