Solvability in a polarized calculus

Author: Please provide author information

Abstract

We investigate the existence of operational characterizations of solvability, i.e. reductions that are normalizing exactly on solvable terms, in calculi with mixed evaluation order (i.e. call-by-name and call-by-value) and pattern-matches. We start by introducing focused call-by-name and call-by-value λ-calculi isomorphic to the intuitionistic fragments of call-by-value and call-by-name $\overline{\lambda}\mu\tilde{\mu}$, relating them to λ-calculi in which solvability has been operationally characterized, and operationally characterizing solvability in them. We then merge both calculi into a polarized one, explain its relation to the previous calculi, describe how the presence of clashes (i.e. pattern-matching failures) affects solvability, and show how the operational characterization can be adapted the a dynamically typed / bi-typed variant of the calculus that eliminates clashes.

2012 ACM Subject Classification Author: Please fill in 1 or more \ccsdesc macro

Keywords and phrases Author: Please fill in \keywords macro

Digital Object Identifier 10.4230/LIPIcs... Xavier Montillet

Introduction

The λ-calculus is a well-known abstraction used to study programming languages. It has two main evaluation strategies: call-by-name (CBN) evaluates subprograms only when they are observed / used, while call-by-value (CBV) evaluates subprograms when they are constructed. Each strategy has its own advantage: CBN ensures that no unnecessary computations are done, while CBV ensures that no computations are duplicated. Somewhat surprisingly, the study of CBV turned out to be more involved than that of CBN, for example requiring computation monads [18, 19] to build models. Some properties of CBN given by Barendregt in 1984 [6] have yet to be adapted to CBV. Levy's call-by-push-value (CBPV) [16, 17] decomposes Moggi's computation monad as an adjunction, subsumes both CBV and CBN and sheds some light on the interactions and differences of both strategies.

Another direction the λ-calculus has evolved in is the computational interpretation of classical logic, with the continuation-passing style translation and Parigot's $\lambda\mu$-calculus [23]. This eventually led to Curien and Herbelin's $\overline{\lambda}\mu\tilde{\mu}$-calculus [10]. An interesting property of $\overline{\lambda}\mu\tilde{\mu}$ is that it resembles both the λ-calculus and the Krivine abstract machine [15], allowing to speak of both the equational theory and the operational semantics. It also sheds more light on the relationship between CBN and CBV: the full calculus is not confluent because of the Lafont critical pair [12], which, when restricted to the intuitionistic fragment becomes

$$\underline{U[T/x]} \triangleleft \underline{\mathsf{let}\ x = T\ \mathsf{in}\ U} \triangleright \mathsf{let}\ x = \underline{T}\ \mathsf{in}\ U$$

where the underlined subterm is the one that the machine is currently trying to evaluate. This is exactly the distinction between CBN (where we substitute T for x immediately), and CBV (where we want to evaluate T before substituting it, and hence move the focus to T). Since CBV is syntactically dual to CBN in $\overline{\lambda}\mu\tilde{\mu}$, the additional difficulty in the study of CBV can be understood as coming from the restriction to the intuitionistic fragment.

Surprisingly, those two lines of work (CBPV and $\overline{\lambda}\mu\tilde{\mu}$) lead to very similar calculi, and both can be combined into Curien, Fiore, and Munch-Maccagnoni's polarized sequent calculus LJ_p^η [9], an intuitionistic variant of (a syntax for) Danos, Joinet and Schellinx's LK_p^η [11]. The main difference between (the abstract machine of) CBPV and LJ_p^η is the same as that of the Krivine abstract machine and the CBN fragment of $\overline{\lambda}\mu\tilde{\mu}$: Subcomputations are

Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

also represented by subcommands / subconfigurations, so that the "abstract machine style" evaluation is no longer restricted to the top-level. The difference between $\bar{\lambda}\mu\tilde{\mu}$ and LJ^{η}_{p} is that instead of begin restricted to a single evaluation strategy (which is necessary in $\bar{\lambda}\mu\tilde{\mu}$ to preserve confluence), both are available, and commands are annotated by a polarity $+$ (for CBV) or $-$ (for CBN) to denote the current evaluation strategy, which removes the Lafont critical pair. The type system also changes from classical logic to intuitionistic logic with explicitly-polarised connectives.

In this article, we introduce an alternative concrete syntax for the untyped but well-polarized intuitionistic fragment of LJ^{η}_{p}. This new syntax, $\underline{\lambda_{\mathrm{p}}}$, is more or less a normal λ-calculus where focus is represented by underlinement. This allows us to widen the audience of this paper by not requiring knowledge of $\bar{\lambda}\mu\tilde{\mu}$.

Solvability

In this article, we use $\underline{\lambda_{\mathrm{p}}}$ to study one of the basic blocks of the theory of the λ-calculus: solvability. A term is *solvable* if there is some way to "use" it that leads to a "result". Solvability plays a central role in the study of the λ-calculus because while it could be tempting to consider λ-terms without a normal form as meaningless, doing so leads to an inconsistent theory. Quoting from [3] (itself quoting from [25]):

> [...] only those terms without normal forms which are in fact unsolvable can be regarded as being "undefined" (or better now: "totally undefined"); by contrast, all other terms without normal forms are at least partially defined. Essentially the reason is that unsolvability is preserved by application and composition [...] which [...] is not true in general for the property of failing to have a normal form.

One of the nice properties of the CBN λ-calculus is that solvability can be operationally characterized: There exists a decidable restriction of the reduction (the head reduction) that is normalizing exactly on solvable terms. This operational characterization is one of the first steps in the study of Böhm trees and observational equivalence. The operational characterization has been extended to CBV [21, 3].

In this article, we replay the proof of [3] in $\underline{\lambda^{\mathrm{pure}}_{\mathrm{n}}}$ and $\underline{\lambda^{\mathrm{pure}}_{\mathrm{v}}}$, the pure call-by-name and call-by-value fragments of $\underline{\lambda^{\mathrm{pure}}_{\mathrm{p}}}$, and then generalize it to $\underline{\lambda^{\mathcal{PN}}_{\mathrm{p}}}$, the dynamically typed / bi-typed variant of $\underline{\lambda^{\mathrm{pure}}_{\mathrm{p}}}$.

Goals

The goals of this article are:

- To give an alternative concrete syntax $\underline{\lambda_{\mathrm{p}}}$ for the well-polarized intuitionistic fragment of LJ^{η}_{p}, that remains readable without any knowledge of $\bar{\lambda}\mu\tilde{\mu}$;
- To convince the reader of the usefulness of $\underline{\lambda_{\mathrm{p}}}$ to study solvability and associated notions, and perhaps get some readers to read this draft[1] that relates $\underline{\lambda_{\mathrm{p}}}$ (in its abstract-machine-like syntax) to CBN and CBV λ-calculi and CBPV;
- To pave the way for the study of Böhm tree and observational equivalence in $\underline{\lambda_{\mathrm{p}}}$, introducing and motivating several notions that will be useful for that purpose;
- To summarize the structure of the proof of operational characterization given in [3].

[1] https://xavier.montillet.ac/drafts/PPDP-2020-submission/

Outline

In Section 1, we recall a few standard definitions, and give a generic theorem that will be used for all proofs of operational characterizations of solvability. In Section 2, we introduce call-by-name and call-by-value focused calculi, and prove that they have an operational characterization of solvability. In Section 3, introduce a polarized focused calculus, and discuss the effect of the presence of clashes on solvability, modify the calculus to remove clashes, and finally operationally characterize solvability in it.

Conventions and notations

In this article, we will describe several calculi, and will use the same conventions for all of them.

Calculi

We write $T[V/x]$ for the capture-avoiding substitution of the free occurrences of x by V in T. We also use contexts $\mathbb{K}$, i.e. expressions (terms, values, ...) with a hole $\square$. We write $\mathbb{K}\boxed{T}$ for the result of plugging T in $\mathbb{K}$, i.e. the result of the *non*-capture-avoiding substitution of the unique occurrence of $\square$ by T in $\mathbb{K}$. Similar constructions in different calculi will be differentiated by adding a symbol: N or n for call-by-name, V or v for call-by-value, p for polarized (or + and − when the polarized calculus contains two variants).

Reductions

We use three reductions: The top-level reduction $\succ$ is used to factor the definitions of the two other reductions. The operational reduction $\triangleright$ is the one that defines the operational semantics of the calculus, and can be defined as the closure or the top-level reduction $\succ$ under a chosen set of contexts, called evaluation contexts and denoted by $\mathbb{E}$. For all the calculi in this paper, the operational reduction $\triangleright$ is deterministic (i.e. $T^1 \triangleleft T \triangleright T^2$ implies $T^1 = T^2$). The strong reduction $\rightharpoonup$ defines the (oriented) equational theory, and is defined as the closure of the top-level reduction $\succ$ under all contexts (i.e. it can reduce anywhere).

We write $\rightsquigarrow$ for an arbitrary reduction (i.e. an arbitrary binary relation whose domain and codomain are equal). Given a reduction $\rightsquigarrow$, we write $\rightsquigarrow^+$ for its transitive closure and $\rightsquigarrow^*$ for its reflexive transitive closure. We say that T $\rightsquigarrow$-reduces to T', written $T \rightsquigarrow T'$, when $(T, T') \in \rightsquigarrow$. Relations will sometimes be used as predicate in which case the second argument is to be understood as existentially quantified (e.g. $T \rightsquigarrow$ means that there exists T' such that $T \rightsquigarrow T'$) unless the relation is striked in which case it should be understood as universally quantified (e.g. $T \not\rightsquigarrow$ means that for all T', $T \not\rightsquigarrow T'$, in other words there exists no T' such that $T \rightsquigarrow T'$). We will say that T is $\rightsquigarrow$-reducible if $T \rightsquigarrow$ and $\rightsquigarrow$-normal otherwise. We will say that T' is a $\rightsquigarrow$-normal form of T if $T \rightsquigarrow^* T' \not\rightsquigarrow$, and that T has an $\rightsquigarrow$-normal form if such a T' exists. If $\rightsquigarrow$ is deterministic, we will say that T $\rightsquigarrow$-converges if it has a normal form, and that it diverges otherwise.

1 Solvability

In this section, we recall a few standard definitions in the pure call-by-name λ-calculus, we which we will call λ_N^{pure}: $T_N, U_N, V_N, W_N ::= x^N \mid \lambda x^N.T_N \mid T_N U_N$. We added N (for call-by-name) subscripts / superscripts everywhere to differentiate it from other calculi that will be introduced. Note that we use V_N and W_N to denote arbitrary terms. As is often done, we write $T_N V_N W_N$ for $(T_N V_N) W_N$. We use several types of contexts (i.e. terms with a hole $\Box$): stacks / weak-head contexts $\mathbb{S}_N ::= \Box V_N^1 \ldots V_N^k$, head contexts $\mathbb{H}_N ::= (\lambda x_1{}^N.\ldots\lambda x_k{}^N.\Box) V_N^1 \ldots V_N^l$, ahead context $\mathbb{A}_N ::= \Box \mid \mathbb{A}_N V_N \mid \lambda x^N.\mathbb{A}_N$ and (strong) contexts $\mathbb{K}_N ::= \Box \mid \lambda x^N.\mathbb{K}_N \mid \mathbb{T}_N U_N \mid T_N \mathbb{U}_N$. We write $\succ$ for the top-level β-reduction $(\lambda x^{\mathrm{n}}.T_N) U_N \succ T_N[U_N/x^N]$. To each type of context, we associate a reduction which is the closure of $\succ$ under those contexts: The operational / weak-head reduction is $\rhd$, the head reduction $\mathrel{\cdots\!\rhd}_{\mathrm{hd}}$, the ahead reduction $\mathrel{\cdots\!\rhd}$ and the strong reduction $\mathrel{-\!\!\rhd}$. We write I_N for $\lambda x^N.x^N$, δ_N for $\lambda x^N.x^N x^N$ and Ω_N for $\delta_N \delta_N$. We use the following definition of solvability, which is easily shown equivalent to the usual one $\lambda_N^{\mathrm{pure}}$ (which can be found, e.g. in [4]):

▶ **Definition 1.** *A term T_N is said to be* solvable *when there exists a variable x^N, a substitution σ_N and a stack $\mathbb{S}_N$ such that $\mathbb{S}_N\boxed{T_N[\sigma_N]} \mathrel{-\!\!\rhd}^* x^N$.*

A nice property of solvability in the call-by-name λ-calculus is that it can be operationally characterized:

▶ **Definition 2.** *A reduction $\rightsquigarrow$ is said to* operationally characterize a set X *of terms when it is deterministic and the set of weakly $\rightsquigarrow$-normalizing terms is X.*

A reduction $\rightsquigarrow$ is said to operationally characterize solvability *when it operationally characterizes the set of solvable terms.*

One of the properties that proofs of this property often involve is sometimes called uniform normalization [14], but we prefer to call it uniqueness of termination behavior[2]:

▶ **Definition 3** (Uniqueness of termination behavior)**.** *A reduction $\rightsquigarrow$ is said to have* uniqueness of termination behavior (UTB) *when weakly $\rightsquigarrow$-normalizing implies strongly $\rightsquigarrow$-normalizing.*

To better understand solvability proofs, it is useful to generalize solvability to an arbitrary reduction $\rightsquigarrow$, with solvability being $\mathrel{-\!\!\rhd}$-solvability:

▶ **Definition 4.** *A term T_N is said to be* $\rightsquigarrow$-solvable *when there exists a variable x^N, a substitution σ_N and a stack $\mathbb{S}_N$ such that $\mathbb{S}_N\boxed{T_N[\sigma_N]} \rightsquigarrow^* x^N$.*

With this definition in mind, a careful reading of [4], combined with a few obvious generalizations and slight reformulations, yields the following properties and theorem (where $\mathrel{\cdots\!\rhd}$ corresponds to their stratified weak reduction $\rightarrow_{\mathrm{sw}}$:

▶ **Proposition 5.** *For any reductions $\mathrel{\cdots\!\rhd}$ and $\mathrel{-\!\!\rhd}$, if* (FactAhead) any reduction $T \mathrel{-\!\!\rhd}^* T'$ can be factorized as $T \mathrel{\cdots\!\rhd}^* \mathrel{-\!\cdot\!\!>}^* T'$ (where $\mathrel{-\!\cdot\!\!>} = \mathrel{-\!\!\rhd} \setminus \mathrel{\cdots\!\rhd}$), *(RedToIAhead) $T \mathrel{-\!\!\rhd} I$ implies $T \mathrel{\cdots\!\rhd} I$, and (InclAhead) $\mathrel{\cdots\!\rhd}^* \subseteq \mathrel{-\!\!\rhd}^*$, then (EqSolAhead) $\mathrel{\cdots\!\rhd}$-solvability and $\mathrel{-\!\!\rhd}$-solvability coincide.*

▶ **Proposition 6.** *For any reduction $\mathrel{\cdots\!\rhd}$, if (NFSol) $\mathrel{\cdots\!\rhd}$-normal terms are solvable, (Disubst) $\mathrel{\cdots\!\rhd}$ is stable under substitution and stacks (i.e. if $T \mathrel{\cdots\!\rhd} T'$ then $T[\sigma] \mathrel{\cdots\!\rhd} T'[\sigma]$ and $\mathbb{S}\boxed{T} \mathrel{\cdots\!\rhd} \mathbb{S}\boxed{T'}$), and (UTB) $\mathrel{\cdots\!\rhd}$ has uniqueness of termination behavior, then (OpCharSelf) $\mathrel{\cdots\!\rhd}$ operationally characterizes $\mathrel{\cdots\!\rhd}$-solvability.*

Combining both properties above, one gets the following theorem:

▶ **Theorem 7.** *For any reductions $\mathrel{\cdots\!\rhd}$ and $\mathrel{-\!\!\rhd}$, if* (FactAhead)*, (RedToVarAhead), (InclAhead), (NFSol), (Disubst), and (UTB) then (OpChar) $\mathrel{\cdots\!\rhd}$ operationally characterizes solvability.*

---

[2] Because the name uniform normalization can easily be misunderstood as implying normalization, which it does not.

The main difficulties when trying to apply this theorem are finding the right $\dashrightarrow$, proving (FactAhead) and proving (NFSol). Proving (UTB) is sometimes also non-trivial. The proof of (FactAhead) became unmanageable for some of the calculi we considered, and we therefore generalize Proposition 5 as follows:

▶ **Proposition 8.** *For any reductions $\triangleright$, $\dashrightarrow$ and $\rightarrowtriangle$, if* (Fact) any reduction $T \rightarrowtriangle^* T'$ can be factorized as $T \triangleright^* \to^* T'$ (where $\to = \rightarrowtriangle \setminus \triangleright$), *(RedToVar) $T \rightarrowtriangle x$ implies $T \triangleright x$, and (Incl) $\triangleright \subseteq \dashrightarrow^* \subseteq \rightarrowtriangle^*$, then (EqSol) $\triangleright$-solvability, $\dashrightarrow$-solvability and $\rightarrowtriangle$-solvability coincide.*

The proof is basically unchanged. Note that replacing all occurrences of $\triangleright$ by $\dashrightarrow$ (and x by I) in Proposition 8 yields Proposition 5, so that Proposition 8 is indeed a generalization of Proposition 5. Combining this with Proposition 6 yields:

▶ **Theorem 9.** *For any reductions $\triangleright$, $\dashrightarrow$ and $\rightarrowtriangle$, if* (Fact)*, (RedToVar), (Incl), (NFSol), (Disubst), and (UTB) then (OpChar) $\dashrightarrow$ operationally characterizes solvability.*

Our experience is that when moving to more larger calculi, $\dashrightarrow$ get very complicated very fast[3], while $\triangleright$ remains relatively simple. Replacing the assumption (FactAhead) by (Fact) is therefore a huge gain. Another very useful advantage of using Proposition 8 is that the proof of (OpChar) can now be split into two relatively independent parts: (EqSol) is mostly independent of the choice of $\dashrightarrow$ with the only assumption on it being (Incl) $\triangleright \subseteq \dashrightarrow^* \subseteq \rightarrowtriangle^*$; while (OpCharSelf) only mentions $\dashrightarrow$. This means that one can prove (EqSol) as soon as the calculus is defined, and then search for the right $\dashrightarrow$ without having to worry about breaking (FactAhead). We recommend looking at Figure 9 and Figure 10 in the appendix, as they should elucidate the structure of the proof of theorem 9.

In the call-by-name λ-calculus, it is well-known [5] that the head reduction $\dashrightarrow_{\mathrm{hd}}$ operationally characterizes solvability. Instead of using $\dashrightarrow_{\mathrm{hd}}$, we prefer using the ahead reduction $\dashrightarrow$ which also characterizes solvability. The main advantage of $\dashrightarrow$ is that the corresponding contexts are stable under composition (i.e. the composition $\mathbb{A}_1\boxed{\mathbb{A}_2}$ of two ahead contexts is always an ahead context, which is not true for head contexts), and its main drawback is that it is not deterministic. This leads to proofs using $\dashrightarrow$ instead of $\dashrightarrow_{\mathrm{hd}}$ being easier to adapt to other calculi (because they do not rely on determinism, and compositionality becomes paramount when the calculus grows in size).

▶ **Theorem 10.** *In λ_N^{pure}, the ahead reduction $\dashrightarrow$ operationally characterizes solvability.*

Proof. We use theorem 9. Among its assumptions: (Subst) and (Fact) are well-known properties; and (Disubst), (RedToVar) and (Incl) are trivial to prove.

- The proof of (UTB) relies on the diamond property: (DP) If $T^l \dashleftarrow T \dashrightarrow T^r$ then either $T^l = T^r$ or $T^l \dashrightarrow T \dashleftarrow T^r$. It is well-known that (DP) implies (UTB).
- The standard proof of (DP) is done as follows: If $T^l \triangleleft T \triangleright T^r$ then $T^l = T^r$ by determinism of $\triangleright$. If$T^l \triangleleft T \dashrightarrow T^r$ then $T^l \dashrightarrow T \triangleleft T^r$ by case analysis on the reduction $T^l \triangleleft T$ and (Disubst). The general case is then by induction on the derivation of both reductions $T^l \dashleftarrow T \dashrightarrow T^r$ until one of the two reductions is an $\triangleright$ reduction or it becomes apparent that the two reductions are applied to disjoint subterms.
- The standard proof of (NFSol), i.e. that $\dashrightarrow$-normal terms are solvable, is as follows. It is easy to prove that $\dashrightarrow$-normal terms T are of the shape $\lambda {x_1}^N.\ldots\lambda {x_k}^N.y^N V_N^1\ldots V_N^l$. Define $\mathrm{o}^l = \lambda {z_1}^N.\ldots\lambda {z_l}^N.z_{l+1}$ where z_{l+1} is a free variable. The idea is to substitute y

[3] Because it has to deal with clashes and reduce several redexes at once in some calculi, as we will see later.

(a) Syntax

Values / terms

$$V_n, W_n, T_n, U_n \;::=\; x^{\mathrm{n}} \mid C_{\rightsquigarrow\mathrm{n}} \mid \lambda x^{\mathrm{n}}.\, C_{\rightsquigarrow\mathrm{n}}$$

Commands

$$C_{\rightsquigarrow\mathrm{n}} \;::=\; \underline{\underline{T_n}\, V_n^1 \dots V_n^k} \mid \underline{\mathsf{let}\; x^{\mathrm{n}} = \underline{T_n}\; \mathsf{in}\; C_{\rightsquigarrow\mathrm{n}}}$$

(b) Stacks and evaluation contexts

Stacks

$$\mathbb{S}_{\mathrm{n}} \;::=\; \Box_n V_n^1 \dots V_n^k$$

Evaluation contexts

$$\mathbb{E}_{\mathrm{n}} \;::=\; \Box_n V_n^1 \dots V_n^k \mid \mathsf{let}\; x^{\mathrm{n}} = \Box_n\; \mathsf{in}\; C_{\rightsquigarrow\mathrm{n}}$$

(c) Definition of $\mathsf{defer}\,(\mathbb{S}_{\mathrm{n}}, C_{\rightsquigarrow\mathrm{n}})$

$$\mathsf{defer}\left(\Box_n V_n^1 \dots V_n^k, \underline{\mathsf{let}\; x^{\mathrm{n}} = \underline{T_n}\; \mathsf{in}\; C_{\rightsquigarrow\mathrm{n}}}\right) = \underline{\mathsf{let}\; x^{\mathrm{n}} = \underline{T_n}\; \mathsf{in}\, \mathsf{defer}\left(\Box_n V_n^1 \dots V_n^k, C_{\rightsquigarrow\mathrm{n}}\right)}$$

$$\mathsf{defer}\left(\Box_n V_n^1 \dots V_n^k, \underline{\underline{T_n}\, W_n^1 \dots W_n^l}\right) = \underline{\underline{T_n}\, W_n^1 \dots W_n^l\, V_n^1 \dots V_n^k}$$

(d) Operational reduction

$$\begin{aligned}
\underline{\underline{C_{\rightsquigarrow\mathrm{n}}}\, V_n^1 \dots V_n^k} \;&\triangleright_\mu\; \mathsf{defer}\,(\Box_n V_n^1 \dots V_n^k, C_{\rightsquigarrow\mathrm{n}}) \\
\underline{\underline{\lambda x^{\mathrm{n}}.\, C_{\rightsquigarrow\mathrm{n}}}\, V_n W_n^1 \dots W_n^k} \;&\triangleright_\to\; \mathsf{defer}\,(\Box_n V_n^1 \dots V_n^k, C_{\rightsquigarrow\mathrm{n}}[T_n/x^{\mathrm{n}}]) \\
\underline{\mathsf{let}\; x^{\mathrm{n}} = \underline{T_n}\; \mathsf{in}\; C_{\rightsquigarrow\mathrm{n}}} \;&\triangleright_{\tilde\mu}\; C_{\rightsquigarrow\mathrm{n}}[T_n/x^{\mathrm{n}}]
\end{aligned}$$

(e) Strong reduction

$$\frac{C_{\rightsquigarrow\mathrm{n}} \triangleright C_{\rightsquigarrow\mathrm{n}}'}{\mathbb{K}_{\mathrm{n}}\boxed{C_{\rightsquigarrow\mathrm{n}}} \to \mathbb{K}_{\mathrm{n}}\boxed{C_{\rightsquigarrow\mathrm{n}}'}}$$

Figure 1 The pure focused call-by-name λ-calculus $\underline{\lambda_{\mathrm{n}}^{\to}}$

by o^l so that the arguments $V_N^1, \dots, V_N^l$ are discarded and we get the z_{l+1}. There are two subcases depending on whether y is equal to one of the or is free in T. In the first case, $y = x_j$ for some j, and the stack $\mathbb{S}_N = \Box W_N^1 \dots W_N^k$ with $W_N^j = \mathrm{o}^l$ allows to conclude: $\mathbb{S}_N\boxed{T_N} \triangleright^* z_{l+1}$. In the second case, y is free in T, in which case the stack $\mathbb{S}_N = \Box W_N^1 \dots W_N^k$ and the substitution $\sigma_N = y^N \mapsto \mathrm{o}^l$ allow to conclude: $\mathbb{S}_N\boxed{T_N[\sigma_N]} \triangleright^* z_{l+1}$.

◀

2 Solvability in focused calculi

2.1 The pure focused call-by-name λ-calculus: $\underline{\lambda_{\mathrm{n}}^{\to}}$

2.1.1 Syntax

We now introduce the pure focused call-by-name λ-calculus, which we call $\underline{\lambda_{\mathrm{n}}^{\mathrm{pure}}}$. It is an alternative concrete syntax for the intuitionistic call-by-name fragment of $\bar\lambda\mu\tilde\mu$. For the pure call-by-name case, using $\underline{\lambda_{\mathrm{n}}^{\mathrm{pure}}}$ is overkill and the usual call-by-name λ-calculus $\lambda_N^{\mathrm{pure}}$ would be enough. We nevertheless use $\underline{\lambda_{\mathrm{n}}^{\mathrm{pure}}}$ to familiarize the reader with focused calculi, because they will helpful for the call-by-value case, and very helpful for the polarized case. There are two kinds of objects in the syntax given in Figure 1a: Terms and commands. If one ignores $\underline{\cdot}$, $\underline{\cdot}$, and the distinction between terms and commands, one gets the usual syntax. Note that any command $C_{\rightsquigarrow\mathrm{n}}$ can be seen as a term, and that any term T_n can be seen as a command $\underline{T_n}$ (which is $\underline{T_n}\, V_n^1 \dots V_n^k$ with $k = 0$). The distinction between a command and a term is that commands are what we reduce while a term is what we substitute for a variable[4]. Commands are similar to those in abstract machines, where $\langle T | \mathbb{K}\rangle$ represents the term $\mathbb{K}\boxed{T}$ where the machine is currently focused on the subterm T. Here, we write $\underline{\mathbb{K}\boxed{\underline{T}}}$ for $\langle T | \mathbb{K}\rangle$,

[4] The terms by which we allow to substitute variables are called values, but in call-by-name all terms are values.

i.e. $\underline{\cdot}$ represents the $\langle$ and $\rangle$, and $\underset{\smile}{\cdot}$ represents the $|$. Just like in abstract machines, the reductions are thought of as interaction between a term and a context, i.e. we do not have $\langle(\lambda x.T)U|\Box\rangle \triangleright \langle T[U/x]|\Box\rangle$ but $\langle(\lambda x.T)|\Box U\rangle \triangleright \langle T[U/x]|\Box U\rangle$. In our syntax, this means not having $\underline{(\lambda x^{\mathrm{n}}.C_{\rightsquigarrow\mathrm{n}})U_n} \triangleright C_{\rightsquigarrow\mathrm{n}}[U/x]$ but instead having $\underline{\underset{\smile}{(\lambda x^{\mathrm{n}}.C_{\rightsquigarrow\mathrm{n}})}\,U_n} \triangleright C_{\rightsquigarrow\mathrm{n}}[U/x]$.

Some contexts will be particularly useful and are therefore given names. Evaluation context $\mathbb{E}_{\mathrm{n}}$ are contexts that can be combined with terms to form commands. More precisely, all commands are of the shape $\underline{\mathbb{E}_{\mathrm{n}}\boxed{T_n}}$, and given any evaluation context $\mathbb{E}_{\mathrm{n}}$ and term T_n, $\underline{\mathbb{E}_{\mathrm{n}}\boxed{T_n}}$ is a command. A stack $\mathbb{S}_{\mathrm{n}}$ is an evaluation context that "can be moved", in much the same way as a value is a term that "can be moved" in the call-by-value λ-calculus. Given a stack $\mathbb{S}_{\mathrm{n}}$ and a command $C_{\rightsquigarrow\mathrm{n}}$, $\mathsf{defer}(\mathbb{S}_{\mathrm{n}}, C_{\rightsquigarrow\mathrm{n}})$ can be though of as a smart way of plugging $C_{\rightsquigarrow\mathrm{n}}$ into $\mathbb{S}_{\mathrm{n}}$. The resulting term will have the same meaning $C_{\rightsquigarrow\mathrm{n}}\boxed{\mathbb{S}_{\mathrm{n}}}$ but may not be strictly equal to it. The idea is to push the stack so that it appears as late as possible in the computation, but before it is needed. For example in $\mathsf{defer}\left(\Box_n V_n, \underline{\mathsf{let}\, x^{\mathrm{n}} = \underset{\smile}{T_n}\ \mathsf{in}\ \underline{\underset{\smile}{\lambda y^{\mathrm{n}}.\, y^{\mathrm{n}}}}}\right)$, we could simply plug the command in the stacks and get $\underline{\underset{\smile}{\left(\mathsf{let}\, x^{\mathrm{n}} = \underset{\smile}{T_n}\ \mathsf{in}\ \underset{\smile}{\lambda y^{\mathrm{n}}.\, y^{\mathrm{n}}}\right)}V_n}$, but the V_n is not needed by the let so there is no point in keeping it here, and we might as well move it further into the computation, which leads to $\underline{\mathsf{let}\, x^{\mathrm{n}} = \underset{\smile}{T_n}\ \mathsf{in}\ \underline{\underset{\smile}{\lambda y^{\mathrm{n}}.\, y^{\mathrm{n}}}\, V_n}}$. This is very much related to commutative cuts[5]. Note that moving the stack in such a way makes the $\underline{\underset{\smile}{\lambda y^{\mathrm{n}}.\, y^{\mathrm{n}}}\, V_n}$ redex apparent, while simply plugging would have lead to this redex being unavailable until the let expression is reduced. In the call-by-name case, this makes the calculus more complex than needed, but in the call-by-value case where some sort of commutative cuts (or other extension) are needed to fully evaluate open terms [2], this will prove very helpful.

An alternative description of the syntax, closer to $\bar{\lambda}\mu\tilde{\mu}$ and more suited for proofs can be found in Figure **??**. More information on how $\underline{\lambda^{\mathrm{pure}}_{\mathrm{n}}}$ is related to $\bar{\lambda}\mu\tilde{\mu}$ can be found in this draft[6], and should help understand why $\mathsf{defer}(\mathbb{S}_{\mathrm{n}}, C_{\rightsquigarrow\mathrm{n}})$ is defined this way (which is that the intuitionistic fragment of $\bar{\lambda}\mu\tilde{\mu}$ has a stack variables $\star$, that $\mathsf{defer}(\mathbb{S}_{\mathrm{n}}, C_{\rightsquigarrow\mathrm{n}})$ corresponds to $C_{\rightsquigarrow\mathrm{n}}[\mathbb{S}_{\mathrm{n}}/\star]$).

From this point on, all numbered definitions, lemmas, propositions and theorems should by default be understood are holding for all subsequent calculi. Proofs will be adapted as needed, and properties that do not hold for all calculi will state explicitly in which calculi they hold. The following lemmas are easily proven by induction:

▶ **Lemma 11.** *The operational reduction $\triangleright$ is disubstitutive: If $C \triangleright C'$ then for any disubstitution φ, $C[\varphi] \triangleright C'[\varphi]$.*

▶ **Lemma 12.** *The strong reduction $\twoheadrightarrow$ is disubstitutive: If $C \twoheadrightarrow C'$ then for any disubstitution φ, $C[\varphi] \twoheadrightarrow C'[\varphi]$.*

▶ **Lemma 13.** *The operational reduction $\triangleright$ is deterministic: If $C^l \triangleleft C \triangleright C^r$ then $C^l = C^r$.*

2.1.2 Solvability

Since we will often use a substitution σ and a stack $\mathbb{S}$ at the same time, we give this kind of pair a name.

[5] But is not exactly the same since it moves the whole stack at once instead of moving arguments one by one, and it can move through several let expressions at once, while commutative cuts typically swap two constructors locally.

[6] https://xavier.montillet.ac/drafts/PPDP-2020-submission/

$$\frac{C_{\rightsquigarrow n} \triangleright C_{\rightsquigarrow n}{}'}{C_{\rightsquigarrow n} \dashrightarrow C_{\rightsquigarrow n}{}'} \qquad \frac{C_{\rightsquigarrow n} \dashrightarrow C_{\rightsquigarrow n}{}'}{\underline{\mathbb{S}_n \boxed{\lambda x^n . C_{\rightsquigarrow n}}} \dashrightarrow \underline{\mathbb{S}_n \boxed{\lambda x^n . C_{\rightsquigarrow n}{}'}}} \qquad \frac{C_{\rightsquigarrow n} \dashrightarrow C_{\rightsquigarrow n}{}'}{\underline{\mathbb{S}_n \boxed{C_{\rightsquigarrow n}}} \dashrightarrow \underline{\mathbb{S}_n \boxed{C_{\rightsquigarrow n}{}'}}}$$

Figure 2 The ahead reduction $\dashrightarrow$ in $\underline{\lambda_n^{\text{pure}}}$

▶ **Definition 14.** *A* disubstitution *is a pair* $(\sigma, \mathbb{S})$ *consisting of a substitution* σ *and as stack* $\mathbb{S}$.

Given a disubstitution $\varphi = (\sigma, \mathbb{S})$, we will write $C[\varphi]$ for $\mathsf{defer}\,(\mathbb{S}, C[\sigma])$.

▶ **Definition 15.** *A disubstitution* φ *is said to* solve *a command* C*, written* $\varphi \vDash C$*, when there exists a variable* x *such that* $C[\varphi] \triangleright^* \underline{x}$*. A command* C *is said to be* solvable*, written* $\exists \vDash C$*, when there exists a disubstitution* φ *that solves it. A term* T *is said to be* solvable *when* $\underline{T}$ *is. An evaluation context* $\mathbb{E}$ *is said to be* solvable *when* $\underline{\mathbb{E}\boxed{\underline{x}}}$ *is for some variable* x.

▶ **Lemma 16.** *(Fact) A sequence of strong reductions* $C \rightharpoonup^* C'$ *can be factorized as* $C \triangleright^* \rightarrow^* C'$ *(where* $\rightarrow \,=\, \rightharpoonup \setminus \triangleright$*).*

Proving factorization $\rightharpoonup^* \subseteq \rightsquigarrow^* (\rightharpoonup \setminus \rightsquigarrow)^*$ for an arbitrary reduction $\rightsquigarrow \subseteq \rightharpoonup$ is highly non-trivial. What makes this factorization easy to prove is that, if we use a well-chosen concrete syntax, the redex that $\triangleright$ reduces is always above all other redexes. Indeed, if we use the abstract-machine-like syntax $\langle T | \mathbb{E} \rangle$, then $\triangleright$ is exactly the top-level reduction. In this syntax, we could use a generic theorem for higher-order rewrite systems proven by Bruggink in [7]:

▶ **Theorem 17** (Theorem 5.5.1 (Standardization Theorem) of [7])**.** *In any local higher-order rewrite system, for every finite reduction, there exists a unique, permutation equivalent, standard reduction. This standard reduction is the same for permutation equivalent reductions.*

If we chose to reduce β-redexes to let-redexes instead of directly substituting, i.e. $\underline{\lambda x^n . C_{\rightsquigarrow n} V_n W_n^1 \ldots W_n^k} \triangleright_{\rightarrow}$ $\mathsf{defer}\left(\Box_n V_n^1 \ldots V_n^k, \underline{\mathsf{let}\, x^n = \underline{T_n}\ \mathsf{in}\, C_{\rightsquigarrow n}}\right)$, which does not change the calculus much, then [1] would most likely apply. Since we refrained both from using the abstract-machine-like syntax (to make the article more accessible), and from decomposing the $\triangleright_{\rightarrow}$ reduction[7], we need to prove factorization by hand. It is nevertheless easily provable using the parallel reduction (see [13, 24]). By Proposition 8, we therefore get the following (because (RedToVar) is trivial, and (Incl) will be once $\dashrightarrow$ is defined in Figure 2):

▶ **Proposition 18.** *A command* C *is solvable if and only if it is* $\dashrightarrow$*-solvable.*

2.1.3 Operational characterization of solvability

The ahead reduction $\dashrightarrow$ is defined in Figure 2. Note that (Incl), i.e. $\triangleright \subseteq \dashrightarrow \subseteq \rightharpoonup$, holds. We now prove the assumptions of theorem **??**.

▶ **Lemma 19.** *The ahead reduction is disubstitutive: For any disubstitution* φ*,* $C \dashrightarrow C'$ *implies* $C[\varphi] \dashrightarrow C'[\varphi]$.

▶ **Lemma 20.** *In* $\underline{\lambda_n^{pure}}$*, the ahead reduction has the diamond property.*

[7] To keep an exact correspondence with the abstract-machine-like calculus, where such a decomposition would induce an arbitrary choice between $\langle \mu\alpha.\langle v | \tilde{\mu}x.c \rangle | s \rangle$ and $\langle v | \tilde{\mu}x.\langle \mu\alpha.c | s \rangle \rangle$.

The standard argument for proving that $\dashrightarrow\!\triangleright$-normal forms are solvable in call-by-name is simply to look at the normal form and immediately deduce a disubstitution that solves it. This would be possible here, but we use a more "small-step" approach that will be easier to generalize.

▶ **Lemma 21.** *In $\underline{\lambda_n^{pure}}$, $\dashrightarrow\!\triangleright$-normal forms are solvable.*

Proof. Define $|C_{\rightsquigarrow n}|_{\text{con}}$ (resp. $|C_{\rightsquigarrow n}|_{\text{des}}$) to be the number of applications (resp. abstractions) in $C_{\rightsquigarrow n}$. We show that if $C_{\rightsquigarrow n} \not\dashrightarrow\!\triangleright$, then there exists a disubstitution φ_n such that $C_{\rightsquigarrow n}[\varphi_n] \triangleright C_{\rightsquigarrow n}' \not\dashrightarrow\!\triangleright$ such that $(|C_{\rightsquigarrow n}|_{\text{con}}, |C_{\rightsquigarrow n}|_{\text{des}}) >_{\text{lex}} (|C_{\rightsquigarrow n}'|_{\text{con}}, |C_{\rightsquigarrow n}'|_{\text{des}})$ (i.e. either $|C_{\rightsquigarrow n}|_{\text{con}} > |C_{\rightsquigarrow n}'|_{\text{con}}$ or $|C_{\rightsquigarrow n}|_{\text{con}} = |C_{\rightsquigarrow n}'|_{\text{con}}$ and $|C_{\rightsquigarrow n}|_{\text{des}} > |C_{\rightsquigarrow n}'|_{\text{des}}$) by case analysis on the shape of $C_{\rightsquigarrow n} = \mathbb{E}_{n}\boxed{\underline{T_n}}$. If $C_{\rightsquigarrow n} = \underline{\lambda x^{n}. C_{\rightsquigarrow n}{}^{2}}$ then $\varphi_n = (\mathrm{Id}, \Box y^{n})$ works. If $C_{\rightsquigarrow n} = \mathbb{S}_{n}\boxed{x^{n} V_n}$ then $\varphi_n = \left(x^{n} \mapsto \lambda\_^{n}.\underline{y^{n}}, \Box\right)$ works.

By iterating this property, we get $C_{\rightsquigarrow n}[\varphi_n] \triangleright C_{\rightsquigarrow n}'$, $C_{\rightsquigarrow n}'[\varphi_n'] \triangleright C_{\rightsquigarrow n}''$ and so on. Since $(|C_{\rightsquigarrow n}|_{\text{con}}, |C_{\rightsquigarrow n}|_{\text{des}})$ strictly decreases and the lexicographical ordering is well-founded, this sequence is necessarily finite. We can therefore take $\psi_n = \cdots \circ \varphi_n' \circ \varphi_n$, and by lemma 11, we get $C_{\rightsquigarrow n}[\psi_n] \triangleright^* C_{\rightsquigarrow n}{}^{\star}$ where $C_{\rightsquigarrow n}{}^{\star} \not\dashrightarrow\!\triangleright$ and $(|C_{\rightsquigarrow n}{}^{\star}|_{\text{con}}, |C_{\rightsquigarrow n}{}^{\star}|_{\text{des}}) = (0, 0)$. The command $C_{\rightsquigarrow n}{}^{\star}$ is therefore a variable $\underline{y^{n}}$ and we are done. ◀

▶ **Theorem 22.** *In $\underline{\lambda_n^{pure}}$, $\dashrightarrow\!\triangleright$ operationally characterizes solvability.*

2.2 The pure focused call-by-value λ-calculus: $\underline{\lambda_{\mathbf{v}}^{\rightarrow}}$

The syntax is the same except that $\mathsf{let}\, x^{v} = \Box_v V_v^1 \ldots V_v^k \,\mathsf{in}\, C_{\rightsquigarrow v}$ is now a stack, and $C_{\rightsquigarrow v}$ is no longer a value. Defer is extended by $\mathsf{defer}\left(\mathsf{let}\, x^{v} = \Box_v V_v^1 \ldots V_v^k \,\mathsf{in}\, C_{\rightsquigarrow v}, \underline{\underline{T_n} W_v^1 \ldots W_v^l}\right) = \underline{\mathsf{let}\, x^{v} = \underline{T_n} W_v^1 \ldots W_v^l V_v^1 \ldots V_v^k \,\mathsf{in}\, C_{\rightsquigarrow v}}$ and $\mathsf{defer}\left(\mathsf{let}\, x^{v} = \Box_v V_v^1 \ldots V_v^k \,\mathsf{in}\, C_{\rightsquigarrow v}, \underline{\mathsf{let}\, x^{n} = \underline{T_n} W_v^1 \ldots W_v^l \,\mathsf{in}\, C_{\rightsquigarrow n}}\right) = \underline{\mathsf{let}\, x^{n} = \underline{T_n} W_v^1 \ldots W_v^l \,\mathsf{in}\, \mathsf{defer}\left(\mathsf{let}\, x^{v} = \Box_v V_v^1 \ldots V_v^k \,\mathsf{in}\, C_{\rightsquigarrow v}, C_{\rightsquigarrow n}\right)}$. Reductions $\triangleright$ and $\rightarrow\!\!\triangleright$ are restricted as usual, i.e. if some term is going to be substituted, then it has to be a value. The ahead reduction is extended by an additional rule:

$$\frac{C_{\rightsquigarrow v} \dashrightarrow\!\triangleright C_{\rightsquigarrow v}'}{\underline{\mathsf{let}\, x^{v} = \mathbb{S}_{v}\boxed{\underline{T_v}} \,\mathsf{in}\, C_{\rightsquigarrow v}} \dashrightarrow\!\triangleright \underline{\mathsf{let}\, x^{v} = \mathbb{S}_{v}\boxed{\underline{T_v}} \,\mathsf{in}\, C_{\rightsquigarrow v}'}}$$

The commutations rules are handled by the $\triangleright_\mu$ reduction. All the lemma are proved using the same techniques except (NFSol), which changes slightly because we have to generalize $C_{\rightsquigarrow n}[\varphi_n] \triangleright C_{\rightsquigarrow n}' \not\dashrightarrow\!\triangleright$ to $C_{\rightsquigarrow v}[\varphi_v] \triangleright\!\dashrightarrow\!\triangleright^* C_{\rightsquigarrow v}' \not\dashrightarrow\!\triangleright$. Indeed, the disubstitution $\varphi_v = \left(x^{v} \mapsto \lambda\_^{v}.\underline{y^{v}}, \Box\right)$ maybe unblock several redexes, for example in $C_{\rightsquigarrow v} = \mathsf{let}\, y^{v} = \underline{x^{v}} V_v \,\mathsf{in}\, \mathsf{let}\, z^{v} = \underline{x^{v}} W_v \,\mathsf{in}\, I$.

▶ **Theorem 23.** *In $\underline{\lambda_v^{pure}}$, $\dashrightarrow\!\triangleright$ operationally characterizes solvability.*

3 Pure polarized solvability

3.1 Calculus

3.1.1 Definition and properties

We not introduce a pure focused λ-calculus that subsumes both call-by-name and call-by-value. Just like the pure call-by-name and call-by-value focused calculi described earlier, it is another syntax for the intuitionistic fragment of an abstract-machine-like calculus: $\mathrm{LJ}_{\mathrm{p}}^{\eta}$ [8]

or $\mathrm{L}_{\mathrm{int}}$ of [20]. Those calculi avoid the Lafont critical pair [12] $C^2[\underline{C^1}/x] \triangleleft \underline{\mathsf{let}\, x = \underline{C^1}\ \mathsf{in}\, C^2} \triangleright \mathsf{defer}\,(\underline{\mathsf{let}\, x = \square\, \mathsf{in}\, C^2}, C^1)$ by adding polarities: $+$ and $-$. The $-$ polarity corresponds to call-by-name and only allows the reduction $C_{\leadsto -}{}^2\big[\underline{C_{\leadsto -}{}^1}/x\big] \triangleleft \underline{\mathsf{let}\, x^- = \underline{C_{\leadsto -}{}^1}\ \mathsf{in}\, C_{\leadsto -}{}^2}$, while the $+$ polarity corresponds to call-by-value and only allows the right reduction $\underline{\mathsf{let}\, x^+ = \underline{C_{\leadsto +}{}^1}\ \mathsf{in}\, C_{\leadsto +}{}^2} \triangleright \mathsf{defer}\left(\mathsf{let}\, x^+ = \square_+\, \mathsf{in}\, C_{\leadsto +}{}^2, C_{\leadsto +}{}^1\right)$. This ensures that $\triangleright$ remains deterministic.

The previously-mentionned calculi were build to study well-typed terms in a classical (i.e. not intuitionistic) setting, and are therefore not perfectly suited for the study of intuitionistic untyped computations. We therefore slightly modify them. We start by taking well-polarized terms, i.e. well-typed terms for the type system where all judgements $T : A_\varepsilon$ are replaced by $T : \varepsilon$. We then restrict to the intuitionistic fragment. Finally, we notice that the set of well-polarized terms is context-free[8], i.e. there exists a context-free grammar that generates them, and therefore that they can be taken as syntax. The resulting syntax can be found in Figure 4a. We will motivate the restriction to well-polarized terms later.

$$\begin{tikzcd} V_+ \arrow[d, phantom, "\cap"] & & V_- \arrow[ll, "\mathsf{unbox}^{\mathrm{P}}"'] \\ T_+ \arrow[urr, "\mathsf{freeze}^{\mathrm{P}}"] & & \end{tikzcd}$$

Figure 3 Shifts

In this calculus, positive values V_+ can be though of as being results, negative term T_- and negative-returning commands $C_{\leadsto -}$ as being computations that will evaluate only if their result is needed, and positive terms T_+ and positive-returning commands $C_{\leadsto +}$ as computations that will evaluate immediately if given the change. Shifts, which allows both polarities to interact, are described in Figure 3. In order to remember the domain and codomain of each shift, one can notice that both shifts inject terms of one polarity into values of the other polarity, and that the $\mathsf{freeze}^{\mathrm{P}}$ shift goes from positive to negative just like with temperatures. The first shift, $\mathsf{freeze}^{\mathrm{P}}(C_{\leadsto +})$, represents a frozen / delayed computation. It is very commonly used in call-by-value programming languages to simulate call-by-name: $\mathsf{freeze}^{\mathrm{P}}(C_{\leadsto +})$ can be though of as being $\lambda().T_+$, and $\mathsf{unfreeze}^{\mathrm{P}}(V_-)$ as being $V_-()$ (where $()$ is the unique inhabitant of the unit type). This amounts to representing a delayed computation of type A as a term of type $\mathsf{unit} \to A$. The second shift, $\mathsf{box}^{\mathrm{P}}(T_-)$, represents the term T_- "marked" as being a result. The corresponding match forces evaluation to a result. By "marking" values and forcing evaluation to "marked" terms before substituting, one can simulate call-by-value in call-by-name. This is somewhat dual to the first shift: $\mathsf{freeze}^{\mathrm{P}}(T_+)$ stops evaluation and $\mathsf{unfreeze}^{\mathrm{P}}(V_-)$ resumes it, while $\mathsf{match}^{\leadsto\varepsilon}\, T_+\, \mathsf{with}\, [\mathsf{box}^{\mathrm{P}}(x^-).C_{\leadsto\varepsilon}]$ forces evaluation until it is stopped by a $\mathsf{box}^{\mathrm{P}}(T_-)$. Through the lens of abstract machines, where a term and a context interact, the shift from T_+ to $\mathsf{freeze}^{\mathrm{P}}(T_+)$ can be though of as giving more power to the context that can now decide when to evaluate T_+, while the shift from $\mathsf{let}\, x^+ = T_+\, \mathsf{in}\, C_{\leadsto\varepsilon}$ to $\mathsf{match}^{\leadsto\varepsilon}\, T_+\, \mathsf{with}\, [\mathsf{box}^{\mathrm{P}}(x^-).C_{\leadsto\varepsilon}]$ can be though of a giving more power to the term that can now decide to return before fully evaluating by boxing the remaining computation. For a detailed description of the relationship between call-by-name, call-by-value, shift, and call-by-push-value, we refer the reader to this draft[9].

An evaluation context is annotated by two polarities, e.g. $\mathbb{E}_{\varepsilon_1 \leadsto \varepsilon_2}$, where the first one ε_1 is the polarity of the input, i.e. of the hole $\square_{\varepsilon_1}$, and the second ε_2 is the polarity of the output, i.e. of $\mathbb{E}_{\varepsilon_1 \leadsto \varepsilon_2}\boxed{\underline{T_{\varepsilon_1}}}$. The fact that $\mathbb{E}_{\varepsilon_1 \leadsto \varepsilon_2}\boxed{\underline{T_{\varepsilon_1}}}$ is always a command $C_{\leadsto \varepsilon_2}$ is not immediately obvious and needs to be proven. In fact, all commands are of this shape. This should not be surprising since we built this calculus as an alternative concrete syntax of an

[8] The reason for this is that instead of having to remember a type, which is an unbounded quantity of information, one only has to remember a polarity, which is a bounded quantity of information.

[9] https://xavier.montillet.ac/drafts/PPDP-2020-submission/

(a) Syntax

$$
\begin{array}{llll}
\multicolumn{4}{l}{\text{Positive values}}\\
& V_+, W_+ & ::= & x^+\\
& & \mid & \mathsf{box}^{\mathsf{p}}(V_-)\\
\multicolumn{4}{l}{\text{Positive terms}}\\
& T_+, U_+ & ::= & V_+ \mid C_{\rightsquigarrow+}\\
\multicolumn{4}{l}{\text{Negative values / terms}}\\
& V_-, W_-, T_-, U_- & ::= & x^- \mid C_{\rightsquigarrow-}\\
& & \mid & \lambda x^+ . C_{\rightsquigarrow-}\\
& & \mid & \mathsf{freeze}^{\mathsf{p}}(C_{\rightsquigarrow+})
\end{array}
$$

$$
\begin{array}{llll}
\multicolumn{4}{l}{+\text{-returning command with a stack}}\\
& D_{\rightsquigarrow+} & ::= & \underline{T_+} \mid \mathsf{let}^{\rightsquigarrow+}\, x^+ = \underline{T_+}\ \mathsf{in}\ C_{\rightsquigarrow+}\\
& & \mid & \mathsf{match}^{\rightsquigarrow+}\, \underline{T_+}\ \mathsf{with}\ [\mathsf{box}^{\mathsf{p}}(x^-).C_{\rightsquigarrow+}]\\
& & \mid & \mathsf{unfreeze}^{\mathsf{p}}(D_{\rightsquigarrow-})\\
\multicolumn{4}{l}{+\text{-returning command}}\\
& C_{\rightsquigarrow+} & ::= & D_{\rightsquigarrow+} \mid \mathsf{let}^{\rightsquigarrow+}\, x^- = \underline{T_-}\ \mathsf{in}\ C_{\rightsquigarrow+}\\
\multicolumn{4}{l}{-\text{-returning command with a stack}}\\
& D_{\rightsquigarrow-} & ::= & \underline{T_-}\\
& & \mid & D_{\rightsquigarrow-} V_+\\
& & \mid & \mathsf{let}^{\rightsquigarrow+}\, x^+ = \underline{T_+}\ \mathsf{in}\ C_{\rightsquigarrow-}\\
& & \mid & \mathsf{match}^{\rightsquigarrow-}\, \underline{T_+}\ \mathsf{with}\ [\mathsf{box}^{\mathsf{p}}(x^+).C_{\rightsquigarrow-}]\\
\multicolumn{4}{l}{-\text{-returning command}}\\
& C_{\rightsquigarrow-} & ::= & C_{\rightsquigarrow-} \mid \mathsf{let}^{\rightsquigarrow-}\, x^- = \underline{T_-}\ \mathsf{in}\ C_{\rightsquigarrow-}
\end{array}
$$

(b) Stacks and evaluation contexts

$$
\begin{array}{llll}
\multicolumn{4}{l}{+\text{-returning positive stacks / evaluation contexts}}\\
& \mathbb{S}_{+\rightsquigarrow+}, \mathbb{E}_{+\rightsquigarrow+} & ::= & \Box_+ \mid \mathsf{let}^{\rightsquigarrow+}\, x^+ = \Box_+\ \mathsf{in}\ C_{\rightsquigarrow+}\\
& & \mid & \mathsf{match}^{\rightsquigarrow+}\, \Box_+\ \mathsf{with}\ [\mathsf{box}^{\mathsf{p}}(x^+).C_{\rightsquigarrow+}]\\
\multicolumn{4}{l}{-\text{-returning positive stacks}}\\
& \mathbb{S}_{-\rightsquigarrow-} & ::= & \Box_-\\
& & \mid & \mathbb{S}_{-\rightsquigarrow-}\boxed{\Box_- V_+}\\
& & \mid & \mathbb{S}_{+\rightsquigarrow-}\boxed{\mathsf{unfreeze}^{\mathsf{p}}(\Box_-)}\\
\multicolumn{4}{l}{-\text{-returning positive evaluation context}}\\
& \mathbb{E}_{-\rightsquigarrow-} & ::= & \mathbb{S}_{-\rightsquigarrow-} \mid \mathsf{let}^{\rightsquigarrow-}\, x^- = \Box_-\ \mathsf{in}\ C_{\rightsquigarrow-}
\end{array}
$$

$$
\begin{array}{llll}
\multicolumn{4}{l}{-\text{-returning positive stacks / evaluation contexts}}\\
& \mathbb{S}_{+\rightsquigarrow-}, \mathbb{E}_{+\rightsquigarrow-} & ::= & \mathsf{let}^{\rightsquigarrow-}\, x^+ = \Box_+\ \mathsf{in}\ C_{\rightsquigarrow-}\\
& & \mid & \mathsf{match}^{\rightsquigarrow-}\, \Box_+\ \mathsf{with}\ [\mathsf{box}^{\mathsf{p}}(x^+).C_{\rightsquigarrow-}]\\
\multicolumn{4}{l}{+\text{-returning negative stacks}}\\
& \mathbb{S}_{-\rightsquigarrow+} & ::= & \\
& & \mid & \\
& & \mid & \mathbb{S}_{+\rightsquigarrow+}\boxed{\mathsf{unfreeze}^{\mathsf{p}}(\Box_-)}\\
\multicolumn{4}{l}{-\text{-returning positive evaluation context}}\\
& \mathbb{E}_{-\rightsquigarrow+} & ::= & \mathbb{S}_{-\rightsquigarrow+} \mid \mathsf{let}^{\rightsquigarrow+}\, x^- = \Box_-\ \mathsf{in}\ C_{\rightsquigarrow+}
\end{array}
$$

(c) Alternative description of commands

$$
\begin{array}{lll@{\qquad}lll}
D_{\rightsquigarrow+} & ::= & \mathbb{S}_{+\rightsquigarrow+}\boxed{\underline{T_+}} \mid \mathbb{S}_{-\rightsquigarrow+}\boxed{\underline{T_-}} & D_{\rightsquigarrow-} & ::= & \mathbb{S}_{+\rightsquigarrow-}\boxed{\underline{T_+}} \mid \mathbb{S}_{-\rightsquigarrow-}\boxed{\underline{T_-}}\\
C_{\rightsquigarrow+} & ::= & \mathbb{E}_{+\rightsquigarrow+}\boxed{\underline{T_+}} \mid \mathbb{E}_{-\rightsquigarrow+}\boxed{\underline{T_-}} & C_{\rightsquigarrow-} & ::= & \mathbb{E}_{+\rightsquigarrow-}\boxed{\underline{T_+}} \mid \mathbb{E}_{-\rightsquigarrow-}\boxed{\underline{T_-}}
\end{array}
$$

$$\mathsf{defer}\left(\mathbb{S}_{\varepsilon_2}, \underline{\mathbb{S}'_{\varepsilon_1\rightsquigarrow\varepsilon_2}\boxed{\underline{C_{\rightsquigarrow\varepsilon_1}}}}\right) = \underline{\mathsf{defer}\left(\mathbb{S}_{\varepsilon_2}, \mathbb{S}'_{\varepsilon_1\rightsquigarrow\varepsilon_2}\right)\boxed{\underline{C_{\rightsquigarrow\varepsilon_1}}}}$$

$$\mathsf{defer}\left(\mathbb{S}_{\varepsilon_2}, \mathsf{let}^{\rightsquigarrow\varepsilon_2}\, x^{\varepsilon_1} = \Box_{\varepsilon_1}\ \mathsf{in}\ C_{\rightsquigarrow\varepsilon_2}\right) = \mathsf{let}^{\rightsquigarrow\varepsilon_2}\, x^{\varepsilon_1} = \Box_{\varepsilon_1}\ \mathsf{in}\ \mathsf{defer}\left(\mathbb{S}_{\varepsilon_2}, C_{\rightsquigarrow\varepsilon_2}\right)$$

$$\mathsf{defer}\left(\mathbb{S}_{\varepsilon}, \mathsf{match}^{\rightsquigarrow\varepsilon}\, \Box_+\ \mathsf{with}\ [\mathsf{box}^{\mathsf{p}}(x^-).C_{\rightsquigarrow\varepsilon}]\right) = \mathsf{match}^{\rightsquigarrow\varepsilon}\, \Box_+\ \mathsf{with}\ [\mathsf{box}^{\mathsf{p}}(x^-).\,\mathsf{defer}\left(\mathbb{S}_{\varepsilon}, C_{\rightsquigarrow\varepsilon}\right)]$$

$$\mathsf{defer}\left(\mathbb{S}_{\varepsilon}, \mathbb{S}_{-\rightsquigarrow\varepsilon}\boxed{\Box_- V_+}\right) = \mathsf{defer}\left(\mathbb{S}_{\varepsilon}, \mathbb{S}_{-\rightsquigarrow\varepsilon}\right)\boxed{\Box_- V_+}$$

$$\mathsf{defer}\left(\mathbb{S}_{\varepsilon}, \mathbb{S}_{+\rightsquigarrow\varepsilon}\boxed{\mathsf{unfreeze}^{\mathsf{p}}(\Box_+)}\right) = \mathsf{defer}\left(\mathbb{S}_{\varepsilon}, \mathbb{S}_{+\rightsquigarrow\varepsilon}\right)\boxed{\mathsf{unfreeze}^{\mathsf{p}}(\Box_+)}$$

(d) Operational reduction

$$
\begin{array}{rcl}
\mathbb{S}_{\varepsilon}\boxed{\underline{C_{\rightsquigarrow\varepsilon}}} & \triangleright_{\mu} & \mathsf{defer}\left(\mathbb{S}_{\varepsilon}, C_{\rightsquigarrow\varepsilon}\right)\\
\mathsf{let}^{\rightsquigarrow\varepsilon_1}\, x^{\varepsilon_2} = \underline{V_{\varepsilon_2}}\ \mathsf{in}\ C_{\rightsquigarrow\varepsilon_1} & \triangleright_{\tilde{\mu}} & C_{\rightsquigarrow\varepsilon_1}[V_{\varepsilon_2}/x^{\varepsilon_2}]\\
\mathsf{match}^{\rightsquigarrow\varepsilon}\, \underline{\mathsf{box}^{\mathsf{p}}(V_-)}\ \mathsf{with}\ [\mathsf{box}^{\mathsf{p}}(x^-).C_{\rightsquigarrow\varepsilon}] & \triangleright_{\Downarrow} & C_{\rightsquigarrow+}[V_-/x^-]\\
\mathbb{S}_-\boxed{\underline{\lambda x^+ . C_{\rightsquigarrow-}}\ V_+} & \triangleright_{\rightarrow} & \mathsf{defer}\left(\mathbb{S}_-, C_{\rightsquigarrow-}[V_+/x^+]\right)\\
\mathbb{S}_+\boxed{\mathsf{unfreeze}^{\mathsf{p}}\left(\underline{\mathsf{freeze}^{\mathsf{p}}(C_{\rightsquigarrow+})}\right)} & \triangleright_{\Uparrow} & \mathsf{defer}\left(\mathbb{S}_+, C_{\rightsquigarrow+}\right)
\end{array}
$$

(e) Notations

$$\underline{\mathsf{unbox}^{\mathsf{p}}\left(\mathbb{S}_{\varepsilon\rightsquigarrow+}\boxed{\underline{T_{\varepsilon}}}\right)} = \underline{\mathsf{match}^{\rightsquigarrow-}\, \mathbb{S}_{\varepsilon\rightsquigarrow+}\boxed{\underline{T_{\varepsilon}}}\ \mathsf{with}\ [\mathsf{box}^{\mathsf{p}}(x^-).x^-]}$$

$$\mathbb{S}_+, \mathbb{E}_+ ::= \mathbb{S}_{+\rightsquigarrow+} \mid \mathbb{S}_{+\rightsquigarrow-} \qquad \mathbb{S}_- ::= \mathbb{S}_{-\rightsquigarrow+} \mid \mathbb{S}_{-\rightsquigarrow-} \qquad \mathbb{E}_- ::= \mathbb{E}_{-\rightsquigarrow+} \mid \mathbb{E}_{-\rightsquigarrow-}$$

Figure 4 The pure focused polarized λ-calculus $\underline{\lambda_{\mathrm{p}}^{\mathrm{pure}}}$

$$
\begin{aligned}
?x^N? &= x^- \\
?\lambda x^N.T_N? &= \lambda y^+.\,\mathsf{match}^{\rightsquigarrow -}\ \underline{y^+}\ \mathsf{with}\left[\mathsf{box}^{\mathrm{p}}(x^-).\underline{?T_N?}\right] \\
?T_N U_N? &= \underline{T_N}\ \mathsf{box}^{\mathrm{p}}(U_N)
\end{aligned}
$$

$$
\begin{aligned}
?x^V?_{\mathrm{val}} &= x^- \\
?\lambda x^V.T_N?_{\mathrm{val}} &= \lambda y^+.\,\mathsf{match}^{\rightsquigarrow -}\ y^+\ \mathsf{with}\left[\mathsf{box}^{\mathrm{p}}(x^-).\mathsf{unbox}^{\mathrm{p}}\left(\mathsf{unfreeze}^{\mathrm{p}}\left(\underline{?T_N?}\right)\right)\right] \\
?V_V?_{\mathrm{term}} &= \mathsf{freeze}^{\mathrm{p}}\left(\underline{\mathsf{box}^{\mathrm{p}}(?V_V?_{\mathrm{val}})}\right) \\
?T_V U_V? &= \mathsf{unbox}^{\mathrm{p}}\left(\mathsf{unfreeze}^{\mathrm{p}}\left(\underline{?T_V?_{\mathrm{term}}}\right)\right)\mathsf{box}^{\mathrm{p}}(?T_N?_{\mathrm{term}})
\end{aligned}
$$

Figure 5 Encoding call-by-name and call-by-value into $\underline{\lambda_{\mathrm{p}}}$

abstract-machine-like calculus where commands of the shape $\langle T_{\varepsilon_1} | \mathbb{E}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}\rangle$ are represented by $\mathbb{E}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}\boxed{T_{\varepsilon_1}}$, and this property simply states that our alternative syntax is indeed equivalent. We use this decomposition very often in proofs.

▶ **Lemma 24.** *For any evaluation context $\mathbb{E}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ and term T_{ε_1}, $\mathbb{E}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}\boxed{T_{\varepsilon_1}}$ is a command $C_{\rightsquigarrow \varepsilon_2}$, and any command $C_{\rightsquigarrow \varepsilon_2}$ has a unique decomposition of the shape $\mathbb{E}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}\boxed{T_{\varepsilon_1}}$.*

3.1.2 Encoding call-by-name and call-by-value

Translations from the call-by-name and call-by-value λ-calculus are described in Figure 5. The encoding of call-by-name corresponds to decomposing the implication call-by-name function space $A \Rightarrow_N B$ as $!A \Rightarrow_p B$. We therefore unbox the argument given to the function, and box the argument in the application. The encoding of call-by-value is more tricky. There is another encoding that sends call-by-value terms to positive terms (which should correspond to decomposing $A \Rightarrow_V B$ as $!(A \Rightarrow_p B)$), but it fails to preserve unsolvability so we use a more complicated one that should correspond to $!A \Rightarrow_p !B$. Some intuition on why this encoding works is given in this draft[10]. For both translations (once we take $\multimap_\mu$-normal forms) we get that both reductions send $\triangleright$ to $\triangleright^+$, and preserve both substitutions and stacks, and hence solvability. Proving directly that they preserve unsolvability is hard because not all disubstitutions in the target are in the image of the translation. Fortunately, we have operational characterizations, so it suffices to show that $\cdot\!\cdot\!\triangleright$ is sent to $\cdot\!\cdot\!\triangleright^+$ through the translation.

▶ **Proposition 25.** *Both translations preserve solvability and unsolvability.*

3.1.3 Normal forms and clashes

Looking at $\triangleright$-normal commands, and using the decomposition $\underline{\mathbb{E}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}\boxed{T_{\varepsilon_1}}}$, one gets the following:

▶ **Lemma 26.** *In $\underline{\lambda_p^{pure}}$, an $\triangleright$-normal command $C_{\rightsquigarrow \varepsilon}$ is of one of the following shapes: $\underline{V_\varepsilon}$, $\mathbb{S}_\varepsilon\boxed{x^\varepsilon}$, $\mathbb{S}_-\boxed{\mathit{freeze}^{\mathrm{p}}(C_{\rightsquigarrow +})\,V_+}$ or $\mathbb{S}_+\boxed{\mathit{unfreeze}^{\mathrm{p}}(\lambda x^+.\,C_{\rightsquigarrow -})}$.*

In an abstract-machine-like syntax this corresponds to $\langle V_\varepsilon | \Box_\varepsilon\rangle$, $\langle x^\varepsilon | \mathbb{S}_\varepsilon\rangle$, $\langle \mathsf{freeze}^{\mathrm{p}}(C_{\rightsquigarrow +}) | \mathbb{S}_-\boxed{\Box_-\,V_+}\rangle$ and $\langle \lambda x^+.\,C_{\rightsquigarrow -} | \mathbb{S}_+\boxed{\mathsf{unfreeze}^{\mathrm{p}}(\Box_-)}\rangle$. The first two are expected since we consider $\underline{V_\varepsilon}$ to be a

[10] https://xavier.montillet.ac/drafts/PPDP-2020-submission/

result, and $\mathbb{S}_\varepsilon\boxed{\underline{x^\varepsilon}}$ is an open term waiting for a substitution to continue evaluating. The last two are interaction between two constructors that were not meant to interact. We will call such terms clashes. We give a more general definition of clash:

▶ **Definition 27.** *A command $C_{\rightsquigarrow\varepsilon}$ is said to be a clash when for all disubstitution φ_ε, $C_{\rightsquigarrow\varepsilon}[\varphi_\varepsilon]$ is $\triangleright$-normal.*

▶ **Lemma 28.** *In $\underline{\lambda_p^{pure}}$, clashes are exactly commands of the shape $\mathbb{S}_-\boxed{\underline{\mathsf{freeze}^{\mathrm{p}}(C_{\rightsquigarrow+})}\,V_+}$ or $\mathbb{S}_+\boxed{\mathsf{unfreeze}^{\mathrm{p}}(\underline{\lambda x^+ .\, C_{\rightsquigarrow-}})}$.*

While clashes are easily characterized, this is much harder for commands that will clash no matter how they are used, for example $\mathbb{K}_1\boxed{\mathbb{K}_2\boxed{T_\varepsilon}}$ where $\mathbb{K}_1 = \underline{\mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{p}}(\underline{x^-})\ \mathsf{in}\ \square_\varepsilon}$ and $\mathbb{K}_2 = \underline{\mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{p}}(\underline{x^-}\,V_+)\ \mathsf{in}\ \square_\varepsilon}$ (where the variable being named _ means that it is not used). The intuition is that if x^- is send to $\mathsf{freeze}^{\mathrm{p}}(U_+)$ then $\underline{x^-}\,V_+$ will clash, and if x^- is send to $\lambda x^+ .\, C_{\rightsquigarrow-}$ then $\mathsf{unfreeze}^{\mathrm{p}}(\underline{x^-})$ will clash. Since both of those terms will be evaluated while evaluating $\mathbb{K}_1\boxed{\mathbb{K}_2\boxed{T_\varepsilon}}$, $\mathbb{K}_1\boxed{\mathbb{K}_2\boxed{T_\varepsilon}}$ is bound to clash (or diverge). We will call such problematic commands implicit clashes. They will make the study of solvability in this calculus more complicated.

3.1.4 The bi-typed variant

With the intuition that $\mathsf{freeze}^{\mathrm{p}}(T_+)$ is $\lambda().\,T_+$, and $\mathsf{unfreeze}^{\mathrm{p}}(V_-)$ is $V_-()$, we remove both $\lambda x^+ .\, C_{\rightsquigarrow-}$ and $\mathsf{freeze}^{\mathrm{p}}(T_+)$, and instead add $\lambda\big< x^+ . C_{\rightsquigarrow-}{}^1 \,|\, \mathsf{freeze}^{\mathrm{p}} . C_{\rightsquigarrow+}{}^2 \big>$ with the following reductions:

$$\begin{array}{rcl} \mathbb{S}_-\boxed{\underline{\lambda\big< x^+ . C_{\rightsquigarrow-}{}^1 \,|\, \mathsf{freeze}^{\mathrm{p}} . C_{\rightsquigarrow+}{}^2 \big>}\,V_+} & \triangleright_{\rightarrow} & \mathsf{defer}\left(\mathbb{S}_-, C_{\rightsquigarrow-}{}^1[V_+/x^+]\right) \\ \mathbb{S}_+\boxed{\mathsf{unfreeze}^{\mathrm{p}}\left(\underline{\lambda\big< x^+ . C_{\rightsquigarrow-}{}^1 \,|\, \mathsf{freeze}^{\mathrm{p}} . C_{\rightsquigarrow+}{}^2 \big>}\right)} & \triangleright_{\Uparrow} & \mathsf{defer}\left(\mathbb{S}_+, C_{\rightsquigarrow+}{}^2\right) \end{array}$$

We call the resulting calculus $\underline{\lambda_{\mathrm{p}}^{\mathcal{PN}}}$. The intuition for this calculus comes from two things. First, models of the untyped λ-calculus correspond to typed models with a unique type, which justifies the bi-typed intuition because there are now two types, one per polarity. Secondly, in dynamically typed programming languages, it is possible to have a pattern match that ranges over values of disjoint types (for example integers and booleans), though this is often expressed as a match on the type followed by a match on the value in the type. In this calculus, if we had pattern-matchable pairs $(V_+ \otimes W_+)$, this would mean having a match $\mathsf{match}^{\rightsquigarrow\varepsilon_2}\ \mathbb{S}_{\varepsilon_1}\boxed{T_{\varepsilon_1}}\mathsf{with}\left[\mathsf{box}^{\mathrm{p}}(x^-).C_{\rightsquigarrow\varepsilon}{}^1 | (y^+ \otimes z^+).C_{\rightsquigarrow\varepsilon}{}^2\right]$ instead of the match for $\mathsf{box}^{\mathrm{p}}(V_-)$ and a separate match for pairs. Although it may not be completely clear in the λ-calculus-like syntax we gave, in thecorresponding abstract-machine-like syntax the idea of having a big pattern-match that ranges over all possible positive value constructors is dual to what we did by introducing $\lambda\big< x^+ . C_{\rightsquigarrow-}{}^1 \,|\, \mathsf{freeze}^{\mathrm{p}} . C_{\rightsquigarrow+}{}^2 \big>$. Having a big patterm-match means that positive stacks can handle any positive value they interact with, and having a "big λ" means that negative values can handle any negative stack they interact with.

▶ **Lemma 29.** *In $\underline{\lambda_p^{\mathcal{PN}}}$, there are no clashes , and $\triangleright$-normal command are of one of the following shapes: $\underline{V_\varepsilon}$ or $\mathbb{S}_\varepsilon\boxed{\underline{x^\varepsilon}}$.*

3.2 Solvability

▶ **Example 30.** Any variable x^ε is solvable. The empty stacks $\square_\varepsilon$ are solvable.

▶ **Lemma 31.** *All clashes are unsolvable.*

$$\frac{C \triangleright C'}{C \dashrightarrow C'} \qquad \frac{T_+ \dashrightarrow T'_+}{\mathbb{S}_+\boxed{\underline{T_+}} \dashrightarrow \mathbb{S}_+\boxed{\underline{T'_+}}}\,T_+ \qquad \frac{\mathbb{S}_+ \dashrightarrow \mathbb{S}'_+}{\mathbb{S}_+\boxed{\underline{V_+}} \dashrightarrow \mathbb{S}'_+\boxed{\underline{V_+}'}} \qquad \frac{V_- \dashrightarrow V'_-}{\mathbb{S}_-\boxed{\underline{V_-}} \dashrightarrow \mathbb{S}_-\boxed{\underline{V'_-}}}$$

$$\frac{\mathbb{E}_- \dashrightarrow \mathbb{E}'_-}{\mathbb{E}_-\boxed{\underline{V_-}} \dashrightarrow \mathbb{E}'_-\boxed{\underline{V_-}}}\,\mathbb{E}_- \qquad \frac{C_{\leadsto\varepsilon} \dashrightarrow C_{\leadsto\varepsilon}'}{\mathsf{as\_term}\,(C_{\leadsto\varepsilon}) \dashrightarrow \mathsf{as\_term}\,(C_{\leadsto\varepsilon}')}\,\mu$$

$$\frac{C_{\leadsto-}^{1} \dashrightarrow C_{\leadsto-}^{3} \qquad C_{\leadsto-}^{2} \dashrightarrow C_{\leadsto-}^{4}}{\lambda\left\langle x^{+}.C_{\leadsto-}^{1} \,\middle|\, \mathsf{freeze}^{\mathrm{p}}.C_{\leadsto-}^{2}\right\rangle \dashrightarrow \lambda\left\langle x^{+}.C_{\leadsto-}^{3} \,\middle|\, \mathsf{freeze}^{\mathrm{p}}.C_{\leadsto-}^{4}\right\rangle} \qquad \frac{\mathbb{S}_- \dashrightarrow \mathbb{S}'_-}{\mathbb{S}_-\boxed{\square_-\,V_+} \dashrightarrow \mathbb{S}'_-\boxed{\square_-\,V_+}}$$

$$\frac{\mathbb{S}_+ \dashrightarrow \mathbb{S}'_+}{\mathbb{S}_+\boxed{\mathsf{unfreeze}^{\mathrm{p}}(\square_-)} \dashrightarrow \mathbb{S}'_+\boxed{\mathsf{unfreeze}^{\mathrm{p}}(\square_-)}}$$

$$\frac{C_{\leadsto\varepsilon_2} \dashrightarrow C_{\leadsto\varepsilon_2}'}{\mathsf{let}^{\leadsto\varepsilon_2}\, x^{\varepsilon_1} = \square_{\varepsilon_1}\ \mathsf{in}\ C_{\leadsto\varepsilon_2} \dashrightarrow \mathsf{let}^{\leadsto\varepsilon_2}\, x^{\varepsilon_1} = \square_{\varepsilon_1}\ \mathsf{in}\ C_{\leadsto\varepsilon_2}'}\,\tilde{\mu}$$

$$\frac{C_{\leadsto\varepsilon} \dashrightarrow C_{\leadsto\varepsilon}'}{\mathsf{match}^{\leadsto\varepsilon}\ \square_+\ \mathsf{with}\,[\mathsf{box}^{\mathrm{p}}(x^-).C_{\leadsto\varepsilon}] \dashrightarrow \mathsf{match}^{\leadsto\varepsilon}\ \square_+\ \mathsf{with}\,[\mathsf{box}^{\mathrm{p}}(x^-).C_{\leadsto\varepsilon}']}$$

Figure 6

For positive terms, solvability can be replaced by a simpler notion, potential valuability, introduced by Paolini and Rocca in [22]:

▶ **Definition 32.** *A command $C_{\leadsto\varepsilon}$ is potentially valuable is there exists a substitution σ such that $C_{\leadsto\varepsilon}[\sigma] \triangleright^* \underline{V_\varepsilon}$. A term T_ε is potentially valuable if $\underline{T_\varepsilon}$ is.*

▶ **Lemma 33.** *Solvable commands are potentially valuable.*

▶ **Lemma 34.** *Any potentially valuable positive term T_+ is solvable.*

In $\underline{\lambda_{\mathrm{p}}^{\mathrm{pure}}}$, operationally characterizing solvability may be possible but would most likely involve proving some kind of separation theorem. Indeed, if we take $\mathbb{K}^1 = \mathsf{let}^{\leadsto\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{p}}(\underline{x^-}\,V_+^1)\ \mathsf{in}\ \square$ and $\mathbb{K}^2 = \mathsf{let}^{\leadsto\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{p}}(\underline{x^-}\,V_+^2 W_+)\ \mathsf{in}\ \square$ then $C_{\leadsto\varepsilon} = \mathbb{K}^1\boxed{\mathbb{K}^2\boxed{\underline{y^\varepsilon}}}$ can be $\dashrightarrow$-normal, while it being solvable depends on the relationship between V_+^1 and V_+^2. If they are equal, $C_{\leadsto\varepsilon}$ is unsolvable because whatever function we substitute x^- by will need to return both a frozen computation and a function when given the same input. However, if we take $V_+^1 = \mathsf{box}^{\mathrm{p}}\left(\mathsf{freeze}^{\mathrm{p}}\left(\underline{V_+^3}\right)\right)$ and $V_+^2 = \mathsf{box}^{\mathrm{p}}\left(\lambda\_^+.\,\underline{\mathsf{freeze}^{\mathrm{p}}\left(\underline{V_+^4}\right)}\right)$, then $\varphi = \left(x^- \mapsto \lambda z^+.\,\underline{\mathsf{unbox}^{\mathrm{p}}(\underline{z^+})}, \square_\varepsilon\right)$ solves it. If V_+^1 and V_+^2 are separable (i.e. there are disubstitutions that send them to distinct variables), then $C_{\leadsto\varepsilon}$ is also solvable. We do not operationally characterize solvability in $\underline{\lambda_{\mathrm{p}}^{\mathrm{pure}}}$ in this article.

3.3 Operational characterization of solvability in $\underline{\lambda_{\mathbf{p}}^{\mathcal{PN}}}$

3.3.1 The ahead reduction

The ahead reduction is given inFigure 6. Note that all commands are either of the shape $\underline{\mathbb{S}_+\boxed{\underline{T_+}}}$ or $\underline{\mathbb{E}_-\boxed{\underline{V_+}}}$. Using this, we define “having the control” as follows:

▶ **Definition 35.** *In a command* $\mathbb{S}_+\boxed{T_+}$*, we say that* $\mathbb{S}_+$ *has the control if* T_+ *is a value, and that* T_+ *has it otherwise. In a command* $\mathbb{E}_-\boxed{V_-}$*, we say that* V_- *has the control if* $\mathbb{E}_-$ *is a stack, and that* V_- *has it otherwise.*

The intuition of is that all operational reductions are of the shape $\mathbb{E}_\varepsilon\boxed{T_\varepsilon} \triangleright C[\varphi]$, where C is a subcommand of either T_ε or $\mathbb{E}_\varepsilon$. In fact, any operational reduction after a disubstitution ψ has a similar property: $\mathbb{E}_\varepsilon\boxed{T_\varepsilon}[\psi] \triangleright C[\varphi]$ where C is a subcommand of either T_ε or $\mathbb{E}_\varepsilon$. The side of the command (which we call side because we are thinking of $\langle T_\varepsilon|\mathbb{E}_\varepsilon\rangle$) that has the control is the one that contains this subcommand C and we can know which one it is before knowing ψ! The intuition of where to reduce is then the following:

"The ahead reduction can always reduce the side that has the control, and can reduce the other side only if it can not be discarded."

Any reduction that follows this has a good chance of operationally characterizing solvability (in the absence of clashes, which need to be handled separately). Note that all V_+ are $\dashrightarrow_{\text{Bad}}$-normal, and this choice was made because positive values can be discarded. Also note that in a command $\mathsf{let}^{\rightsquigarrow\varepsilon}\, x^- = \underline{T_-}\, \mathsf{in}\, C_{\rightsquigarrow\varepsilon}$, you can not reduce the T_-, again because it could be discarded. In a classical version on this calculus, one would be able to build terms $\mathsf{magic}\,(C)$ that discard stacks and then compute some other command C, i.e. $\mathbb{S}\boxed{\mathsf{magic}\,(C)} \triangleright C$, and negative stacks that do not have the control therefore would be $\dashrightarrow_{\text{Bad}}$-normal[11]. Here, we are in an intuitionistic calculus, so stacks are never discarded, and we can therefore allow reducing them even when they do not have the control. In fact, not only can they not be discarded, but when moved by defer, they will be moved to somewhere where $\dashrightarrow_{\text{Bad}}$ can reach them:

▶ **Lemma 36.** *If* $\mathbb{S}_\varepsilon \dashrightarrow \mathbb{S}'_\varepsilon$ *then* $\mathit{defer}(\mathbb{S}_\varepsilon, C_{\rightsquigarrow\varepsilon}) \dashrightarrow \mathit{defer}(\mathbb{S}'_\varepsilon, C_{\rightsquigarrow\varepsilon})$*.*

Our syntax does not distinguish between a command $C_{\rightsquigarrow\varepsilon}$ and the same command seen as a term $\mathsf{as\_term}\,(C_{\rightsquigarrow\varepsilon})$, but we made that coercion explicit in the rules. The intuition of why we reduce both commands in parallel in $\lambda\langle x^+.C_{\rightsquigarrow -}{}^1|\mathsf{freeze}^{\mathrm{P}}.C_{\rightsquigarrow -}{}^2\rangle$ is that we want to preserver (Disubst) and (UTB). In $\underline{\lambda\langle x^+.C_{\rightsquigarrow -}{}^1|\mathsf{freeze}^{\mathrm{P}}.C_{\rightsquigarrow -}{}^2\rangle}$, if $\dashrightarrow$ only reduced one side, by disubstitutivity, we could defer a stack that interacts with the other side, so that a $\triangleright$ step could erase the $\dashrightarrow$ reduction step, and this would break (UTB). We now prove that $\dashrightarrow$ operationally characterizes solvability. The proof of (NFSol) just needs $|\cdot|_{\text{con}}$ to be extended to count applications, unfreeze and matches, and $|\cdot|_{\text{des}}$ to count both λ-abstractions, freeze and box. The idea is that $|\cdot|_{\text{con}}$ counts value constructors, while $|\cdot|_{\text{des}}$ counts stack contructors. Note that if your disubstitution is a stack, after reduction, there will be one less value constructor. If the disubstitutions is a substitution however, it will add an arbitrary number of value constructors, while removing only one stack constructor. This is why we use $(\,|\cdot|_{\text{des}}, |\cdot|_{\text{con}}\,)$ and not $(\,|\cdot|_{\text{con}}, |\cdot|_{\text{des}}\,)$.

The proof of (UTB) uses a somewhat unexpected property: $\triangleleft\!-\!\cdots\!\!\triangleright$ is a bisimulation[12] for $\dashrightarrow$, i.e. if $C^l \triangleleft\!-\!\cdots\!\!\triangleright\!\dashrightarrow C^{rr}$ then $C^l \triangleleft\!-\!\cdot\!\triangleleft\!-\!\cdots\!\!\triangleright C^{rr}$. This property arises naturally when trying to prove that the synchronized product[13] of two abstract rewriting systems that have the (DP) has (UTB).

[11] Which is expected because $\mathbb{S}_-\boxed{x^-}$ would be solved by $x^- \mapsto \mathsf{magic}\,(\underline{y^\varepsilon})$

[12] Usually, the definition of bisimulation has two parts, but since $\triangleleft\!-\!\cdots\!\!\triangleright$ is symmetric, we do not need the second one.

[13] The synchronized product of $(\mathcal{A}_1, \rightsquigarrow_1)$ and $(\mathcal{A}_2, \rightsquigarrow_2)$ is $(\mathcal{A}_1 \times \mathcal{A}_2, \rightsquigarrow_3)$ where $(a_1, a_2) \rightsquigarrow_3 (a'_1, a'_2)$ is defined as $a_1 \rightsquigarrow_1 a'_1$ and $a_2 \rightsquigarrow_2 a'_2$.

▶ **Theorem 37.** *In $\underline{\lambda_p^{\mathcal{PN}}}$, $\dashrightarrow$ operationally characterizes solvability.*

Conclusion

While based on calculi geared towards typing and classical logic, the calculus $\mathrm{L}_\mathrm{p}^{\mathcal{PN}}$ has shown to be useful to study solvability, and given how regular η-conversion rules look in it, we believe that it will prove very useful for the study of observational equivalence too. The alternative λ-calculus-like syntax $\underline{\lambda_\mathrm{p}^{\mathcal{PN}}}$, however has proven hard to work with (for us), because the underlinements and defer, while necessary to faithfully represent $\mathrm{L}_\mathrm{p}^{\mathcal{PN}}$, are very easy to forget or misplace. We hope that it nevertheless served its purpose: making $\mathrm{L}_\mathrm{p}^{\mathcal{PN}}$ more accessible.

The ideas that we would like the reader to take home from this article are: the notion of "having the control"; the use of disubstitutions; the idea of making the calculus dynamically typed / bi-typed calculi to remove clashes; and the idea of splitting the proof of the operational characterization on solvability into two very distinct parts.

References

[1] Beniamino Accattoli. "An Abstract Factorization Theorem for Explicit Substitutions". In: *23rd International Conference on Rewriting Techniques and Applications (RTA'12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan.* 2012, pp. 6–21. DOI: `10.4230/LIPIcs.RTA.2012.6`. URL: `https://doi.org/10.4230/LIPIcs.RTA.2012.6`.

[2] Beniamino Accattoli and Giulio Guerrieri. "Open Call-by-Value (Extended Version)". In: *CoRR* abs/1609.00322 (2016). URL: `http://arxiv.org/abs/1609.00322`.

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

[4] Beniamino Accattoli and Luca Paolini. "Call-by-Value Solvability, Revisited". In: *Functional and Logic Programming.* Ed. by Tom Schrijvers and Peter Thiemann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 4–16. ISBN: 978-3-642-29822-6.

[5] H.P. Barendregt. *The lambda calculus: its syntax and semantics.* Studies in logic and the foundations of mathematics. North-Holland, 1984. ISBN: 9780444867483. URL: `https://books.google.fr/books?id=eMtTAAAAYAAJ`.

[6] Hendrik Pieter Barendregt. *The lambda calculus - its syntax and semantics.* Vol. 103. Studies in logic and the foundations of mathematics. North-Holland, 1985. ISBN: 978-0-444-86748-3.

[7] Harrie Jan Sander Bruggink. "Equivalence of Reductions in Higher-Order Rewriting". PhD thesis. Utrecht University, 2008. ISBN: 978-90-393-4817-8. URL: `https://dspace.library.uu.nl:8443/bitstream/handle/1874/27575/bruggink.pdf?sequence=1`.

[8] Pierre-Louis Curien, Marcelo P. Fiore, and Guillaume Munch-Maccagnoni. "A theory of effects and resources: adjunction models and polarised calculi". In: *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016.* Ed. by Rastislav Bodik and Rupak Majumdar. ACM, 2016, pp. 44–56. ISBN: 978-1-4503-3549-2. DOI: `10.1145/2837614.2837652`. URL: `http://doi.acm.org/10.1145/2837614.2837652`.

[9] Pierre-Louis Curien, Marcelo Fiore, and Guillaume Munch-Maccagnoni. "A Theory of Effects and Resources: Adjunction Models and Polarised Calculi". In: *Proc. POPL*. 2016. DOI: 10.1145/2837614.2837652.

[10] Pierre-Louis Curien and Hugo Herbelin. "The duality of computation". In: *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000*. SIGPLAN Notices 35(9). ACM, 2000, pp. 233–243. ISBN: 1-58113-202-6. DOI: http://doi.acm.org/10.1145/351240.351262.

[11] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. "A new deconstructive logic: linear logic". In: *Journal of Symbolic Logic* 62.3 (1997), pp. 755–807. DOI: 10.2307/2275572.

[12] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types*. USA: Cambridge University Press, 1989. ISBN: 0521371813.

[13] Giulio Guerrieri, Luca Paolini, and Simona Ronchi Della Rocca. "Standardization and Conservativity of a Refined Call-by-Value lambda-Calculus". In: *CoRR* abs/1611.07255 (2016). URL: http://arxiv.org/abs/1611.07255.

[14] Zurab Khasidashvili and Mizuhito Ogawa. "Perpetuality and Uniform Normalization". In: *Algebraic and Logic Programming, 6th International Joint Conference, ALP '97 - HOA '97, Southampton, U.K., Spetember 3-5, 1997, Proceedings*. Ed. by Michael Hanus, Jan Heering, and Karl Meinke. Vol. 1298. Lecture Notes in Computer Science. Springer, 1997, pp. 240–255. ISBN: 3-540-63459-2. DOI: 10.1007/BFb0027014. URL: https://doi.org/10.1007/BFb0027014.

[15] Jean-Louis Krivine. "A call-by-name lambda-calculus machine". In: *Higher Order Symbolic Computation* 20 (2007), pp. 199–207. URL: https://hal.archives-ouvertes.fr/hal-00154508.

[16] Paul Blain Levy. *Call-By-Push-Value: A Functional/Imperative Synthesis*. Vol. 2. Semantics Structures in Computation. Springer, 2004. ISBN: 1-4020-1730-8.

[17] Paul Blain Levy. "Call-by-push-value: Decomposing call-by-value and call-by-name". In: *Higher-Order and Symbolic Computation* 19.4 (Dec. 2006), pp. 377–414. ISSN: 1573-0557. URL: https://doi.org/10.1007/s10990-006-0480-6.

[18] 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*. IEEE Computer Society, 1989, pp. 14–23. DOI: 10.1109/LICS.1989.39155. URL: https://doi.org/10.1109/LICS.1989.39155.

[19] Eugenio Moggi. "Notions of Computation and Monads". In: *Inf. Comput.* 93.1 (1991), pp. 55–92. DOI: 10.1016/0890-5401(91)90052-4. URL: https://doi.org/10.1016/0890-5401(91)90052-4.

[20] Guillaume Munch-Maccagnoni and Gabriel Scherer. "Polarised Intermediate Representation of Lambda Calculus with Sums". In: *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*. 2015, pp. 127–140. DOI: 10.1109/LICS.2015.22. URL: http://dx.doi.org/10.1109/LICS.2015.22.

[21] Luca Paolini and Simona Ronchi Della Rocca. "Call-by-value Solvability". In: *ITA* 33.6 (1999), pp. 507–534. DOI: 10.1051/ita:1999130. URL: http://dx.doi.org/10.1051/ita:1999130.

[22] Luca Paolini and Simona Ronchi Della Rocca. "Call-by-value Solvability". In: *RAIRO - Theoretical Informatics and Applications* 33.6 (1999), pp. 507–534. DOI: 10.1051/ita:1999130.

[23] Michel Parigot. "$\lambda\mu$-Calculus: An algorithmic interpretation of classical natural deduction". In: *Logic Programming and Automated Reasoning*. Ed. by Andrei Voronkov. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 190–201. ISBN: 978-3-540-47279-7.

[24] Masako Takahashi. "Parallel Reductions in lambda-Calculus". In: *Inf. Comput.* 118.1 (1995), pp. 120–127. DOI: 10.1006/inco.1995.1057. URL: https://doi.org/10.1006/inco.1995.1057.

[25] Christopher P. Wadsworth. "The Relation Between Computational and Denotational Properties for Scott's $\mathrm{D}_{\mathrm{infty}}$-Models of the Lambda-Calculus". In: *SIAM J. Comput.* 5.3 (1976), pp. 488–521. DOI: 10.1137/0205036. URL: https://doi.org/10.1137/0205036.

(a) Syntax

Terms / values:

$$T_N, U_N, V_N, W_N \quad ::= \quad x^N \mid \lambda x^N.T_N \mid T_N U_N$$

(b) Top-level reduction $\succ$

$$(\lambda x^{\mathrm{n}}.T_N)U_N \quad \succ \quad T_N[U_N/x^N]$$

(c) Contexts

Stacks / operational contexts:

$$\mathbb{S}_N \quad ::= \quad \square V_N^1 \dots V_N^k$$

Head contexts:

$$\mathbb{H}_N \quad ::= \quad (\lambda x_1{}^N \dots \lambda x_k{}^N.\square)V_N^1 \dots V_N^l$$

Ahead contexts:

$$\mathbb{A}_N \quad ::= \quad \square \mid \mathbb{A}_N V_N \mid \lambda x^N.\mathbb{A}_N$$

(Strong) contexts:

$$\mathbb{K}_N \quad ::= \quad \square \mid \lambda x^N.\mathbb{K}_N \mid \mathbb{T}_N U_N \mid T_N \mathbb{U}_N$$

(d) Reductions

Operational / weak head reduction $\triangleright$:

$$\frac{T_N \succ T'_N}{\mathbb{S}_N\boxed{T_N} \triangleright \mathbb{S}_N\boxed{T'_N}}$$

Head reduction $\dashrightarrow_{\mathrm{hd}}$:

$$\frac{T_N \succ T'_N}{\mathbb{H}_N\boxed{T_N} \dashrightarrow_{\mathrm{hd}} \mathbb{H}_N\boxed{T'_N}}$$

Ahead reduction $\dashrightarrow$:

$$\frac{T_N \succ T'_N}{\mathbb{A}_N\boxed{T_N} \dashrightarrow \mathbb{A}_N\boxed{T'_N}}$$

Strong reduction $\rightarrow$:

$$\frac{T_N \succ T'_N}{\mathbb{K}_N\boxed{T_N} \rightarrow \mathbb{K}_N\boxed{T'_N}}$$

Figure 7 The pure call-by-name λ-calculus $\lambda_{\mathrm{n}}^{\mathrm{pure}}$

A Solvability

▶ **Example 38.**

$$I_N \stackrel{\text{def}}{=} \lambda x^N.x^N \qquad K_N \stackrel{\text{def}}{=} \lambda x^N.\lambda y^N.x^N \qquad \delta_N \stackrel{\text{def}}{=} \lambda x^N.x^N x^N \qquad \Omega_N \stackrel{\text{def}}{=} \delta_N \delta_N \triangleright \Omega_N$$

$$\delta_N^{T_N} \stackrel{\text{def}}{=} \lambda x^{\mathrm{n}}.T_N(x^N x^N) \qquad \Omega_N^{T_N} \stackrel{\text{def}}{=} \delta_N^{T_N}\delta_N^{T_N} \triangleright T_N \Omega_N^{T_N} \qquad Y_N \stackrel{\text{def}}{=} \lambda x^N.\Omega_N^{x^N}$$

Proof of theorem 9

A.0.1 Proving that $\dashrightarrow$-solvability is equivalent to $\rightarrow$-solvability

Proof of Proposition 8. ◀

- FactToVar : Suppose that $T \rightarrow^* x$. By (Fact), $T \triangleright^* T' \rightarrow^n x$ for some $n \in \mathbb{N}$. By (RedToVar), there is no T'' such that $T'' \rightarrow x$, so that $n = 0$ and $T' = x$. We can therefore conclude that $T \triangleright^* x$.

- EqSol : Two of the implications are given by (Incl). The remaining one is (FactToVar).

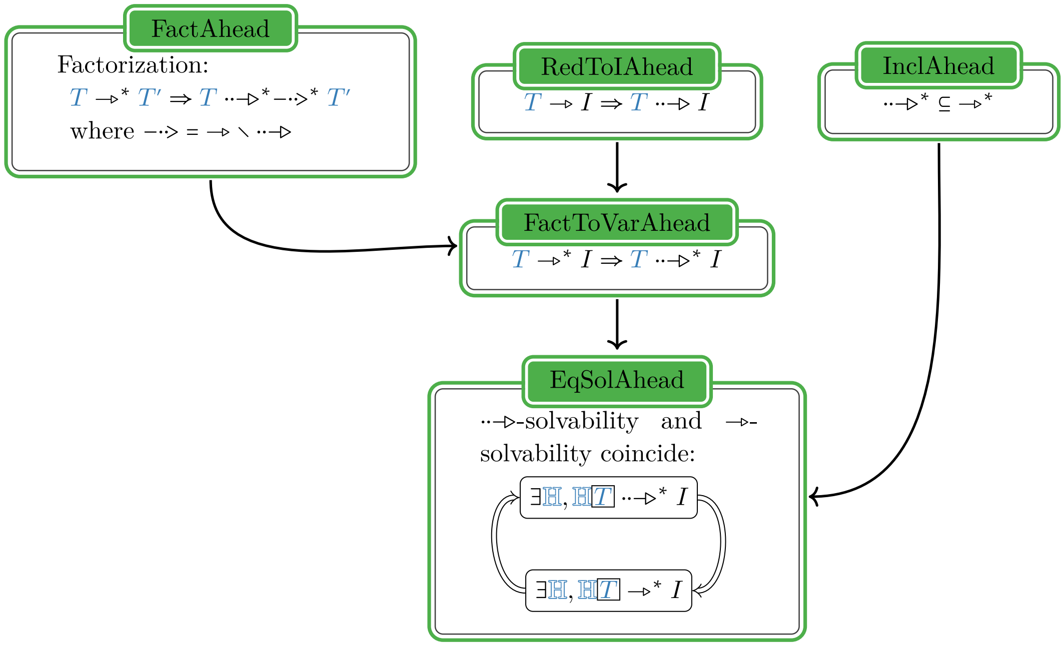

Figure 8 Equivalence of solvability definitions (from [**AccPao12**]) - Proposition 5

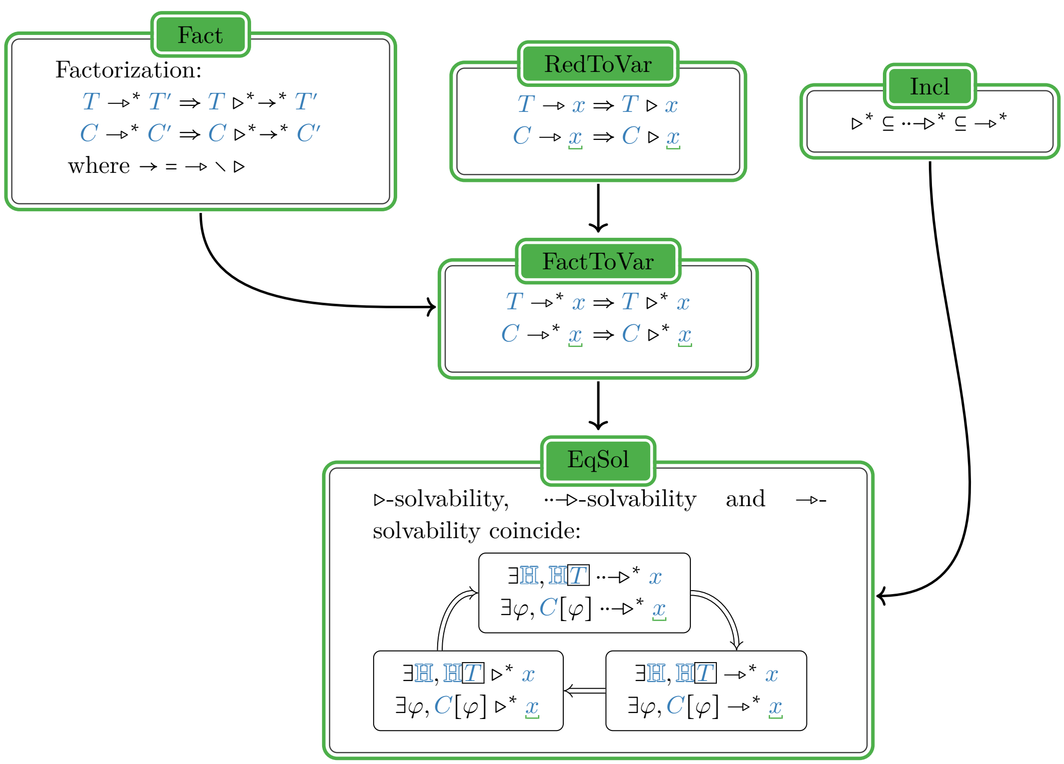

Figure 9 Equivalence of solvability definitions - Proposition 8

A.1 Proving that $\dashrightarrow$ operationally characterizes $\dashrightarrow$-solvability

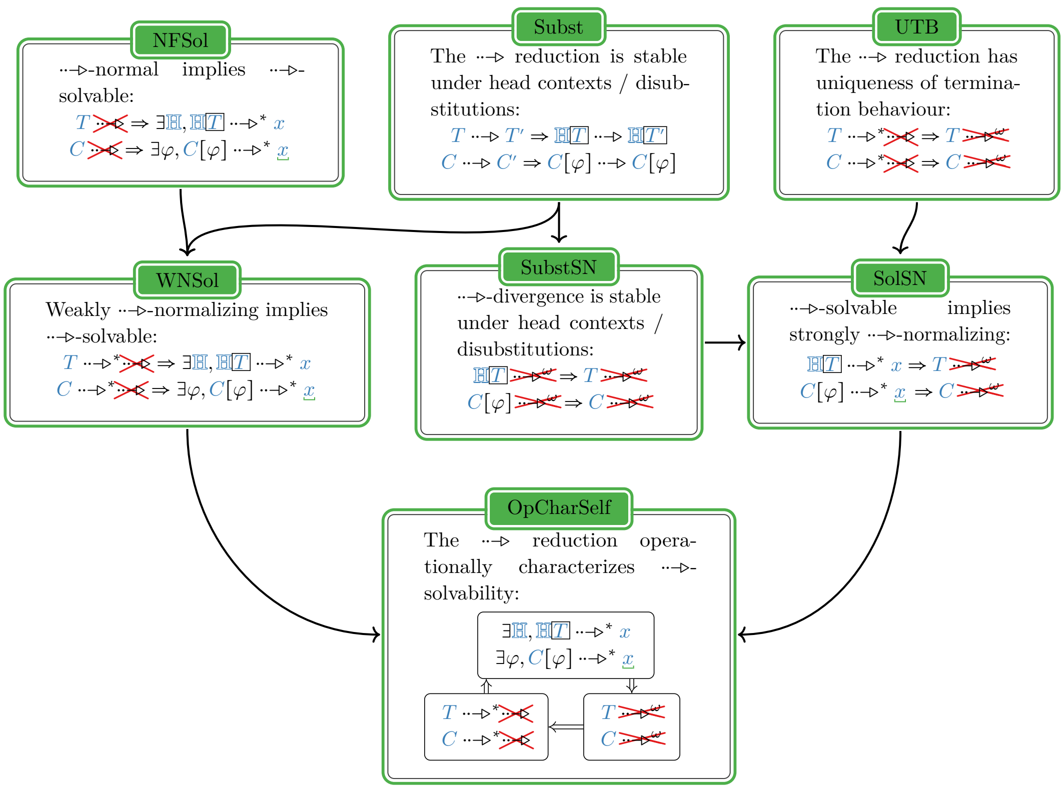

Figure 10 Operational characterization of self-solvability - Proposition 6

Proof of Proposition 6. Intermediate lemmas are described in Figure 10. ◀

- WNSol : Suppose that $T \dashrightarrow^* T' \not\dashrightarrow$. Since $T \not\dashrightarrow$, by (NFSol), there exists $\mathbb{H}$ such that $\mathbb{H}\boxed{T'} \dashrightarrow^* I$. By (Subst), we have $\mathbb{H}\boxed{T} \dashrightarrow^* \mathbb{H}\boxed{T'}$ and we can therefore conclude that $\mathbb{H}\boxed{T} \dashrightarrow^* I$.
- SubstSN : The contrapositive is a corollary of (Subst).
- SolSN : If $\mathbb{H}\boxed{T} \dashrightarrow^* I$, since $I \not\dashrightarrow$, by (UTB), we have $\mathbb{H}\boxed{T} \not\dashrightarrow^{\omega}$. By (SubstSN), we can therefore conclude that $T \not\dashrightarrow^{\omega}$.
- OpCharSelf : (WNSol) and (SolSN) give two of the implications, and the third one (i.e. strongly-normalizing implies weakly-normalizing) is easy: Perform arbitrary $\dashrightarrow$ reduction steps until a normal form is reached (and one is eventually reached because the term is strongly normalizing).

A.2 Proving uniqueness of termination behaviour

A sequence of properties that imply (UTB) are given in Figure 11. For the call-by-name λ-calculus, $\dashrightarrow_{\mathrm{hd}}$ is deterministic, which immediately implies (UTB). As we progress towards more complex calculi, some of those properties will no longer hold, and we will therefore have to prove a lower one directly, which is harder. (Det), (DP) and (UTB) are well-known

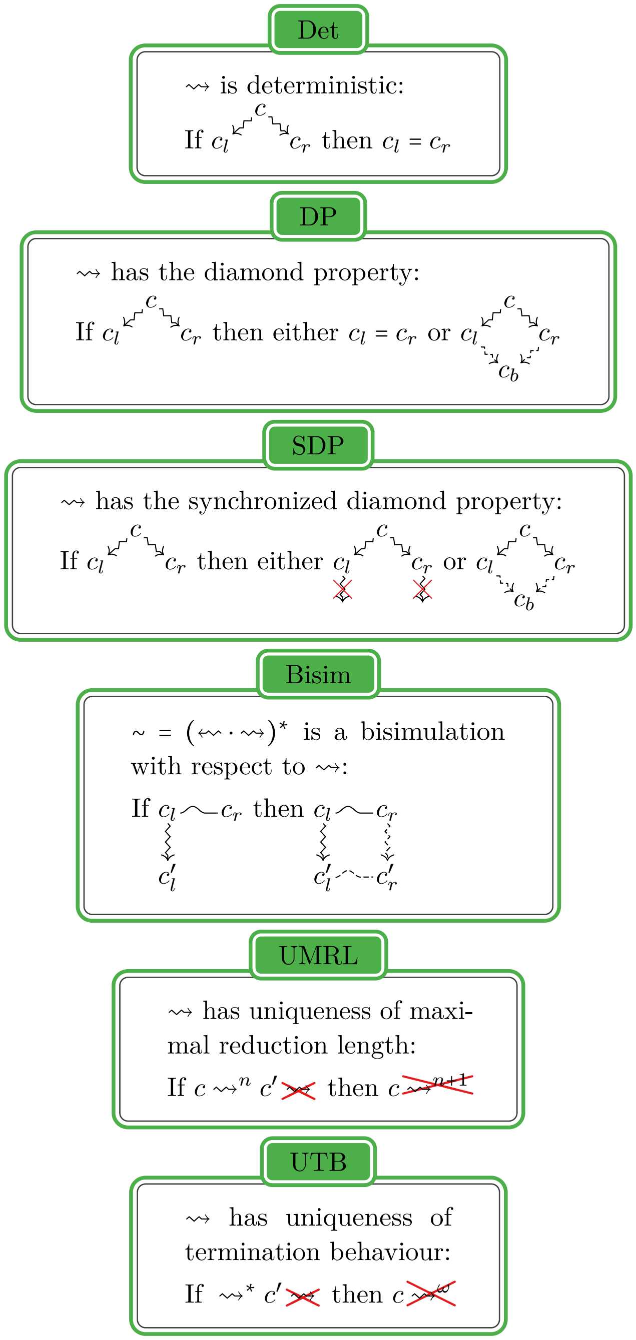

Figure 11 Properties implying uniqueness of termination behavior

properties. (SDP) is what one gets on the synchronized product[14] of two abstract rewriting systems that have (DP). (Bisim) arises naturally when trying to prove that (SDP) implies (UMRL), and our intuition of $\sim$ is that it respects a very strong notion of observational equivalence that has the number of reduction steps as an invariant. (UMRL) states that all maximal reduction (whether finite or infinite) have the same length.

B Solvability in focused calculi

Proof of lemma 19. By induction on the derivation of $C \dashrightarrow C'$. The base case $C \triangleright C'$ is lemma 11. ◀

Proof of lemma 20. By induction on the derivation of the reductions. The only non-trivial cases are $\mathsf{defer}\,(\mathbb{S}_{\mathrm{n}}, C_{\rightsquigarrow\mathrm{n}}) \triangleleft \underline{\mathbb{S}_{\mathrm{n}}\boxed{C_{\rightsquigarrow\mathrm{n}}}} \dashrightarrow \underline{\mathbb{S}_{\mathrm{n}}\boxed{C_{\rightsquigarrow\mathrm{n}}'}}$ and $\mathsf{defer}\,(\mathbb{S}_{\mathrm{n}}, C_{\rightsquigarrow\mathrm{n}}[V_n/x^{\mathrm{n}}]) \triangleleft \underline{\mathbb{S}_{\mathrm{n}}\boxed{(\lambda x^{\mathrm{n}}.\, C_{\rightsquigarrow\mathrm{n}})\, V_n}} \dashrightarrow \underline{\mathbb{S}_{\mathrm{n}}\boxed{(\lambda x^{\mathrm{n}}.\, C_{\rightsquigarrow\mathrm{n}}')\, V_n}}$, both of which are handled via lemma 19. ◀

C Pure polarized solvability

Proof of lemma 24. This lemma is easily proven by proving the same thing for $\mathbb{S}_{\varepsilon_1\rightsquigarrow\varepsilon_2}\boxed{T_{\varepsilon_1}}$ and $D_{\rightsquigarrow\varepsilon_2}$ (by case analysis on the polarities and induction), and then noting that it works for the only remaining case. The only induction hypothesis that needs to be strengthened is to prove that $\mathbb{S}_{-\rightsquigarrow-}\boxed{T_-}$ is always a $D_{\rightsquigarrow-}$, which needs to be stenghened to $\mathbb{S}_{-\rightsquigarrow-}\boxed{D_{\rightsquigarrow-}'}$ is always a $D_{\rightsquigarrow-}$. ◀

Proof of lemma 26. We start by using the decomposition of $C_{\rightsquigarrow\varepsilon_1}$ as $\underline{\mathbb{E}_{\varepsilon_1\rightsquigarrow\varepsilon_2}\boxed{T_{\varepsilon_1}}}$.

We now show that any $\triangleright$-normal command is of the shape $\underline{\mathbb{S}_{\varepsilon_1\rightsquigarrow\varepsilon_2}\boxed{V_{\varepsilon_1}}}$ by contradiction and case analysis on ε_1. If $\varepsilon_1 = -$, then the term T_- is necessarily a value V_-, and the only way for the evaluation context $\mathbb{E}_{-\rightsquigarrow\varepsilon_2}$ to not be a stack $\mathbb{S}_{-\rightsquigarrow\varepsilon_2}$ is to be of the shape $\mathbb{E}_{-\rightsquigarrow\varepsilon_2} = \mathsf{let}^{\rightsquigarrow\varepsilon_2}\, x^- = \square_- \;\mathsf{in}\; C_{\rightsquigarrow\varepsilon_2}$, so that $\underline{\mathbb{E}_{-\rightsquigarrow\varepsilon_2}\boxed{T_-}} = \mathsf{let}^{\rightsquigarrow\varepsilon_2}\, x^- = V_- \;\mathsf{in}\; C_{\rightsquigarrow\varepsilon_2} \triangleright_{\tilde{\mu}} C_{\rightsquigarrow\varepsilon_1}[V_-/x^-]$ and we can conclude that $C_{\rightsquigarrow\varepsilon_1}$ is not $\triangleright$-normal. Dually, if $\varepsilon_1 = +$, then the evaluation context $\mathbb{E}_{+\rightsquigarrow\varepsilon_2}$ is necessarily a stack $\mathbb{S}_{+\rightsquigarrow\varepsilon_2}$, and the only way for the term T_+ to not be a value V_+ is to be of the shape $T_+ = C_{\rightsquigarrow+}$, so that $\underline{\mathbb{E}_{+\rightsquigarrow\varepsilon_2}\boxed{T_+}} = \underline{\mathbb{S}_{+\rightsquigarrow\varepsilon_2}\boxed{C_{\rightsquigarrow+}}} \triangleright_\mu \mathsf{defer}\,(\mathbb{S}_\varepsilon, C_{\rightsquigarrow\varepsilon})$ and we can conclude that $C_{\rightsquigarrow\varepsilon_1}$ is not $\triangleright$-normal.

We now show that amond commands of the shape $\underline{\mathbb{S}_{\varepsilon_1\rightsquigarrow\varepsilon_2}\boxed{V_{\varepsilon_1}}}$, the only $\triangleright$-normal ones are of the shape $\mathbb{S}_-\boxed{\mathsf{freeze}^{\mathrm{p}}(C_{\rightsquigarrow+})\, V_+}$ or $\mathbb{S}_+\boxed{\mathsf{unfreeze}^{\mathrm{p}}(\lambda x^+.\, C_{\rightsquigarrow-})}$. This is done by case analysis on the polarity ε_1 and then the syntax of $\mathbb{S}_{\varepsilon_1\rightsquigarrow\varepsilon_2}$ and V_{ε_1}. ◀

Proof. lemma 28It is immediate that commands of this shape are clashes. To show that all clashes are of this shape, notice that by taking φ_ε to be the identity, we get $C_{\rightsquigarrow\varepsilon} \not\triangleright$ so that $C_{\rightsquigarrow\varepsilon}$ is of one of the four shapes given in the previous lemma. It is easy to find a disubstitution φ_ε such that $C_{\rightsquigarrow\varepsilon}[\varphi_\varepsilon] \triangleright$ if $C_{\rightsquigarrow\varepsilon}$ is of the shape $\langle V_\varepsilon | \square_\varepsilon \rangle$, $\langle x^\varepsilon | \mathbb{S}_\varepsilon \rangle$ which allows to conclude. ◀

Proof of lemma 33. We have $C_{\rightsquigarrow\varepsilon}[\varphi_\varepsilon] \triangleright^* \underline{x}$ with $\varphi_\varepsilon = (\sigma, \mathbb{S}_\varepsilon)$. Since $C_{\rightsquigarrow\varepsilon}[\varphi_\varepsilon]$ is weakly $\triangleright$-normalizing, and hence strongly $\triangleright$-normalizing by lemma 13, so is $C_{\rightsquigarrow\varepsilon}[\sigma]$ by lemma 11. We therefore have $C_{\rightsquigarrow\varepsilon}'$ such that $C_{\rightsquigarrow\varepsilon}[\sigma] \triangleright^* C_{\rightsquigarrow\varepsilon}' \not\triangleright$. If $C_{\rightsquigarrow\varepsilon}' = \underline{x}$, we are done. Otherwise, we

[14] The synchronized product of $(\mathcal{A}_1, \rightsquigarrow_1)$ and $(\mathcal{A}_2, \rightsquigarrow_2)$ is $(\mathcal{A}_1 \times \mathcal{A}_2, \rightsquigarrow_3)$ where $(a_1, a_2) \rightsquigarrow_3 (a_1', a_2')$ is defined as $a_1 \rightsquigarrow_1 a_1'$ and $a_2 \rightsquigarrow_2 a_2'$.

necessarily have $\mathbb{S}_\varepsilon\boxed{C'_{\rightsquigarrow\varepsilon}} \not\triangleright$. This implies that $C'_{\rightsquigarrow\varepsilon}$ can be neither a clash, nor of the shape $\mathbb{S}_\varepsilon\boxed{\underline{x^\varepsilon}}$. By the characterization of $\triangleright$-normal forms it is therefore of the shape $C'_{\rightsquigarrow\varepsilon} = \underline{V_\varepsilon}$, and $C_{\rightsquigarrow\varepsilon}$ is therefore potentially valuable. ◀

Proof of lemma 34. We have $\underline{T_+[\sigma]} \triangleright^* \underline{V_+}$. Take $\varphi_+ = \sigma, \mathsf{let}^{\rightsquigarrow\varepsilon}\, x^+ = \square \,\mathsf{in}\, \underline{y^\varepsilon}$ where $x^+ \neq y^\varepsilon$. We have $\underline{T_+}[\varphi_+] = \underline{\mathsf{let}^{\rightsquigarrow\varepsilon}\, x^+ = \underline{T_+[\sigma]} \,\mathsf{in}\, \underline{y^\varepsilon}} \triangleright^* \underline{\mathsf{let}^{\rightsquigarrow\varepsilon}\, x^+ = \underline{V_+} \,\mathsf{in}\, \underline{y^\varepsilon}} \triangleright \underline{y^\varepsilon}$. ◀

Regarding the proof above, the reader may wonder if $\mathsf{let}^{\rightsquigarrow\varepsilon}\, x^+ = \square \,\mathsf{in}\, \underline{y^\varepsilon}$ should be considered to be a contexts that "effectively uses its hole", since it seems to extract no information from the term plugged in its hole. To answer this, notice that evaluating $\mathsf{let}^{\rightsquigarrow\varepsilon}\, x^+ = \underline{T_+} \,\mathsf{in}\, \underline{y^\varepsilon}$, will also evaluate $\underline{T_+}$. This means that $\mathsf{let}^{\rightsquigarrow\varepsilon}\, x^+ = \underline{T_+} \,\mathsf{in}\, \underline{y^\varepsilon}$ reduces to $\underline{y^\varepsilon}$ if and only if the evaluation of $\underline{T_+}$ terminates, so that even though the information is discarded by returning $\underline{y^\varepsilon}$, the information "$\underline{T_+}$ terminates" has been extracted from the term that was placed in the hole.

There is another, perhaps more convincing, way to look at this: considering that $\mathsf{let}^{\rightsquigarrow\varepsilon}\, x^+ = \square \,\mathsf{in}\, C_{\rightsquigarrow\varepsilon}$ "effectively uses its hole" is expected to be admissible, i.e. disallowing such contexts in the definition of solvability should leave the set of solvable commands unchanged. The idea is that one can replace $\mathbb{S}^1_+ = \mathsf{let}^{\rightsquigarrow\varepsilon}\, x^+ = \square_+ \,\mathsf{in}\, C_{\rightsquigarrow\varepsilon}$ by $\mathbb{S}^2_+ = \mathsf{match}^{\rightsquigarrow\varepsilon}\, \square_+ \,\mathsf{with}\, [\mathsf{box}^{\mathrm{P}}(y^-).C_{\rightsquigarrow\varepsilon}[\mathsf{box}^{\mathrm{P}}(y^-)/x^+]]$ in the disubstitution φ_ε. We do not prove this here as it would involve proving that the η-conversion $\mathbb{S}_+ =_\eta \mathsf{match}^{\rightsquigarrow\varepsilon}\, \square_+ \,\mathsf{with}\, [\mathsf{box}^{\mathrm{P}}(y^-).\mathbb{S}_+\boxed{\mathsf{box}^{\mathrm{P}}(y^-)}]$ respects observational equivalence in this pure calculus, which is non-trivial and left as further work. To give some intuition, we nevertheless adapt the proof that all potentially valuable term T_+ are solvable so as to not use $\mathsf{let}^{\rightsquigarrow\varepsilon}\, x^+ = \square \,\mathsf{in}\, C_{\rightsquigarrow\varepsilon}$. If V_+ is a variable z^+, then $\mathbb{S}_+ = \square_+$ solves it. If V_+ is not a variable then it is of the shape $\mathsf{box}^{\mathrm{P}}(z^-)$, so that $\mathbb{S}_+ = \mathsf{match}^{\rightsquigarrow +}\, \square \,\mathsf{with}\, [\mathsf{box}^{\mathrm{P}}(x^-).\underline{y^+}]$ solves it. In other words, a potentially valuable term T_+ is solvable, not because its result V_+ can be discarded by $\mathsf{let}^{\rightsquigarrow\varepsilon}\, x^+ = \square \,\mathsf{in}\, \underline{y^+}$, but because variables are solvable, and all other positive values have a constructor at their root, so that the corresponding match solves them.

Proof of lemma 31. Suppose by contradiction that a clash $C_{\rightsquigarrow\varepsilon}$ is solved by a disubstitution φ_ε, i.e. $C_{\rightsquigarrow\varepsilon}[\varphi_\varepsilon] \triangleright^* \underline{x^\varepsilon}$. Since $C_{\rightsquigarrow\varepsilon}[\varphi_\varepsilon] \not\triangleright$, we would necessarily have $C_{\rightsquigarrow\varepsilon}[\varphi_\varepsilon] = \underline{x^\varepsilon}$. The only way for this equality to hold is that $C_{\rightsquigarrow\varepsilon}$ is of the shape $\underline{y^\varepsilon}$, which is absurd because $\underline{y^\varepsilon}$ is not a clash. ◀

Proof of lemma 36. By induction on $C_{\rightsquigarrow\varepsilon}$. ◀

▶ **Lemma 39.** *(NFSol) If C is $\dashrightarrow_{Unsol}$-normal then C is solvable.*

Proof of lemma 39. If C were unsolvable, we would have $C \in \mathrm{Unsol}$ and hence $C \dashrightarrow_{\mathrm{Unsol}} C$. ◀

▶ **Lemma 40.** *(Disubst) The ahead reduction $\dashrightarrow_{Bad}$ is disubstitutive: If $C \dashrightarrow_{Bad} C'$ then $C[\varphi] \dashrightarrow_{Bad} C'[\varphi]$.*

Proof of lemma 40. By induction on C. The base cases, which correspond to the two first rules defining $\dashrightarrow_{\mathrm{Bad}}$, use the disubstitutivity of $\triangleright$ and the fact that Bad is closed under disubstitutions. ◀

▶ **Lemma 41.** *(DP) The ahead reduction $\dashrightarrow_{Bad}$ has the diamond property: If $C^l \dashleftarrow_{Bad} C \dashrightarrow_{Bad} C^r$ then either $C^l = C^r$ or $C^l \dashrightarrow_{Bad}\dashleftarrow_{Bad} C^r$.*

Proof of lemma 41. By case analysis on the reduction $C^l \triangleleft C$ and $C \dashrightarrow_{\mathrm{Bad}} C^r$, one gets that $C^l \triangleleft C \dashrightarrow_{\mathrm{Bad}} C^r$ implies $C^l \dashrightarrow_{\mathrm{Bad}} \triangleleft C^r$:

- If $C^l \triangleleft C \dashrightarrow_{\mathrm{Bad}} C$ with $C \in \mathrm{Bad}$ then $C^l \in \mathrm{Bad}$ so $C^l \dashrightarrow_{\mathrm{Bad}} C^l \triangleleft C^r$.
- If $C^l \triangleleft C \triangleright C^r$ then $C^l = C^r$ by determinism of $\triangleright$.
- All other cases are handled as follows: We look at what happens to the redex reduced by $C \dashrightarrow_{\mathrm{Bad}} C^r$ through the $C^l \triangleleft C$ reduction. For most case, the redex will not be impacted by the reduction $C^l \triangleleft C$ and commutation is either trivial, or uses the fact that Bad is closed under disubstitutions if the $C \dashrightarrow_{\mathrm{Bad}} C^r$ relies on some subcommand being in Bad. The only interesting cases arise when the reduction $C \dashrightarrow_{\mathrm{Bad}} C^r$ happens in a stack $\mathbb{S}$ such that get deferred in C^l, and those cases are handled by lemma 36.

◀

▶ Remark 42. $\varphi = \left(x^- \mapsto \lambda z^+.\,\mathsf{unbox}^{\mathrm{P}}(z^+), \square_\varepsilon\right)$ solves $C_{\rightsquigarrow\varepsilon} = \mathbb{K}^1\boxed{\mathbb{K}^2\boxed{y^\varepsilon}}$, where $\mathbb{K}^1 = \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{P}}(x^-\,V_+^1)\ \mathsf{in}\ \square$, $\mathbb{K}^2 = \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{P}}(x^-\,V_+^2 W_+)\ \mathsf{in}\ \square$, $V_+^1 = \mathsf{box}^{\mathrm{P}}\left(\mathsf{freeze}^{\mathrm{P}}\left(V_+^3\right)\right)$ and $V_+^2 = \mathsf{box}^{\mathrm{P}}\left(\lambda\_^+.\,\mathsf{freeze}^{\mathrm{P}}\left(V_+^4\right)\right)$

(where we assume that the two occurrences of x^- are the only ones because otherwise the command would not fit within the page):

$$
\begin{array}{rcl}
C_{\rightsquigarrow\varepsilon}[\varphi] & = & \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{P}}\left(\left(\lambda z^+.\,\mathsf{unbox}^{\mathrm{P}}(z^+)\right)V_+^1\right)\ \mathsf{in}\ \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{P}}\left(\left(\lambda z^+.\,\mathsf{unbox}^{\mathrm{P}}(z^+)\right)V_+^2 W_+\right)\ \mathsf{in}\ y^\varepsilon \\
& \triangleright & \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{P}}\left(\mathsf{unbox}^{\mathrm{P}}\left(V_+^1\right)\right)\ \mathsf{in}\ \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{P}}\left(\left(\lambda z^+.\,\mathsf{unbox}^{\mathrm{P}}(z^+)\right)V_+^2 W_+\right)\ \mathsf{in}\ y^\varepsilon \\
& \triangleright & \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{P}}\left(\mathsf{freeze}^{\mathrm{P}}\left(V_+^3\right)\right)\ \mathsf{in}\ \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{P}}\left(\left(\lambda z^+.\,\mathsf{unbox}^{\mathrm{P}}(z^+)\right)V_+^2 W_+\right)\ \mathsf{in}\ y^\varepsilon \\
& \triangleright & \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = V_+^3\ \mathsf{in}\ \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{P}}\left(\left(\lambda z^+.\,\mathsf{unbox}^{\mathrm{P}}(z^+)\right)V_+^2 W_+\right)\ \mathsf{in}\ y^\varepsilon \\
& \triangleright & \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{P}}\left(\left(\lambda z^+.\,\mathsf{unbox}^{\mathrm{P}}(z^+)\right)V_+^2 W_+\right)\ \mathsf{in}\ y^\varepsilon \\
& \triangleright & \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{P}}\left(\mathsf{unbox}^{\mathrm{P}}\left(V_+^2\right)W_+\right)\ \mathsf{in}\ y^\varepsilon \\
& \triangleright & \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{P}}\left(\left(\lambda\_^+.\,\mathsf{freeze}^{\mathrm{P}}\left(V_+^4\right)\right)W_+\right)\ \mathsf{in}\ y^\varepsilon \\
& \triangleright & \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = \mathsf{unfreeze}^{\mathrm{P}}\left(\mathsf{freeze}^{\mathrm{P}}\left(V_+^4\right)\right)\ \mathsf{in}\ y^\varepsilon \\
& \triangleright & \mathsf{let}^{\rightsquigarrow\varepsilon}\ \_^+ = V_+^4\ \mathsf{in}\ y^\varepsilon \\
& \triangleright & y^\varepsilon
\end{array}
$$