# Solvability in a polarized calculus


## Xavier Montillet

Inria, LS2N CNRS, Nantes, France

Xavier.Montillet@Inria.fr



## Abstract

We investigate 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. To that end, we generalize a polarized abstract-machine-like calculus. We then operationally characterize solvability in several versions of the calculus (classical, pure intuitionistic, ...). In doing so, we illustrate that our calculus is well suited for the study of solvability, that clashes (i.e. pattern-matching failures) are no longer a problem in a polarized calculus, and that operationally characterizing solvability in a classical calculus is easier than in an intuitionistic one. We also show that the main remaining obstacle to the characterization in the full calculus is decidability of separability for "normal-enough" terms.



**2012 ACM Subject Classification**

**Keywords and phrases** call-by-push-value, sequent calculus, solvability

**Digital Object Identifier** 10.4230/LIPIcs...


## Introduction

The $\lambda$-calculus is a well-known abstraction used to study programming languages. It has two distinct evaluation strategies: *call-by-name* (CBN) evaluates things only when they are observed / used, while *call-by-value* (CBV) evaluates things when they are constructed. Both strategies have advantages: 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 [20, 21] to build models. Some properties of CBN, given in [6] in 1984, have yet to be adapted to CBV. *Call-by-push-value* (CBPV) [19] subsumes both CBV and CBN and sheds some light on the interactions and differences of both strategies.

Another direction the $\lambda$-calculus has evolved in is the computational interpretation of classical logic, with the continuation-passing style translation and the $\lambda\mu$ calculus [27]. This eventually led to the $\overline{\lambda}\mu\tilde{\mu}$ calculus [9], which instead of having natural deduction as type system, has the sequent calculus. An interesting property of $\overline{\lambda}\mu\tilde{\mu}$ is that it resembles both the $\lambda$ calculus and the Krivine abstract machine [17], 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 [15]

$$c_1\,[\tilde{\mu}x.c_2/\alpha] \lhd \langle \mu\alpha.c_1 \mid\mid \tilde{\mu}x.c_2 \rangle \rhd c_2\,[\mu\alpha.c_1/x]$$

where $\mu\alpha.c_1$ represents "the result of running the computation $c_1$" and and $\tilde{\mu}x.c_2$ represents the context $\mathsf{let}\, x = \square\, \mathsf{in}\, c_2$, so that the critical pair can be reformulated (if we restrict ourselves to the intuitionistic fragment) as

$$\mathsf{let}\, x = \underline{M_1}\, \mathsf{in}\, M_2 \lhd \underline{\mathsf{let}\, x = M_1\, \mathsf{in}\, M_2} \rhd \underline{M_2\,[M_1/x]}$$

(where the underlined subterm is the one that the machine is currently trying to evaluate). This is exactly the distinction between CBV (where we want to evaluate $M_1$ before substituting it), and CBN (where we substitute it immediately). 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 (as illustrated in Section 5).

Surprisingly, those two lines of work (CBPV and $\overline{\lambda}\mu\tilde{\mu}$) lead to very similar calculi (especially if one looks at the abstract machine of CBPV), and both can be combined into a polarized sequent calculus $\mathrm{LJ}_p^\eta$ [8], an intuitionistic variant of (a syntax for) Danos, Joinet and Schellinx's $\mathrm{LK}_p^\eta$ [11]. The difference between (the abstract machine of) CBPV and $\mathrm{LJ}_p^\eta$ is the same as that of the Krivine abstract machine and the CBN fragment of $\overline{\lambda}\mu\tilde{\mu}$: Subcomputations are also represented by commands / configurations, so that the "abstract machine style" evaluation is no longer restricted to the top-level. The difference between $\overline{\lambda}\mu\tilde{\mu}$ and $\mathrm{LJ}_p^\eta$ is that instead of allowing just one evaluation strategy, both are allowed, and commands are annotated by a polarity $+$ (for CBV) or $-$ (for CBN) to denote the current evaluation strategy. The type system also changes from classical logic to intuitionistic logic with explicitly-polarised connectives.

In this article, we use a slight variation of $\mathrm{LJ}_p^\eta$ which we will call $\mathcal{L}$ here, the main difference being that the calculus is untyped but well-polarized. This calculus inherits many of the advantages of $\overline{\lambda}\mu\tilde{\mu}$: it is abstract-machine-like so that weak head evaluation is just top-level reduction; commuting conversions are built-in and give rise to a confluent reduction; classical logic is built-in but it is easy to restrict to the intuitionistic fragment; CBN and CBV are dual; applicative contexts can be represented by stacks and plugging a term in an applicative context can therefore be seen a substituting a stack for a stack variable. It also inherits many of the advantages of CBPV: It subsumes CBN and CBV and allows mixing both evaluation strategies; it has nice models; and nice $\eta$-conversion laws. The additional restriction to well-polarized terms restricts the possible shapes of *clashes* (pattern-matching failures). It also makes the "dynamically typed" variant (in which pattern matches match over all constructors) clashless.

In order to illustrate the usefulness of the $\mathcal{L}$ calculusIn this article, we use $\mathcal{L}$ to study one of the basic blocks of the theory of the $\lambda$-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 $\lambda$-calculus because while it could be tempting to consider $\lambda$-terms without a normal form as meaningless, doing so leads to an inconsistent theory. Quoting from [5] (itself quoting from [28]):

> [...] 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 $\lambda$-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 [26, 5].

In this article, we extend this proof to $\mathcal{L}$. This allows us to illustrate how having an abstract-machine-like calculus simplifies the proof (because the weak head reduction is the top-level reduction, plugging in an applicative contexts is substituting a stack, and we can often divide by 2 the number of cases by symmetry), that the difficulty of CBV comes from the restriction to the intuitionistic fragment, and that in a polarized calculus all problems due to clashes also appear in the presence of additives (i.e. types with more than one constructor).

## Outline

In Section 1, we introduce our variation of the $\mathrm{LJ}_p^\eta$ calculus: the $\mathcal{L}$ calculus. In Section 2, we define solvability in $\mathcal{L}$, and reduce proving that a reduction operational characterizes solvability to simpler properties (with a proof heavily inspired from [5]). In Section 3, we define the ahead reduction, parameterized by a set of bad commands. In Section 4, we prove that any solvable command is strongly ahead normalizing, independently of the bad set, and for all fragments of the calculus. In Section 5, we prove that in some fragments of the calculus, there exists a set of bad commands such that the induced reduction is decidable, and any weakly ahead normalizing command is solvable.

Readers familiar with the $\lambda$-calculus, but unfamiliar with $\overline{\lambda}\mu\tilde{\mu}$ and / or CBPV and hence with $\mathcal{L}$, should be able understand most of the intuition and the skeleton of most proofs without understanding Section 1. However, it is much more convenient to do actual proofs in the $\mathcal{L}$ calculus, and understanding details of the proofs will therefore require understanding $\mathcal{L}$.

# 1 Polarized calculus

Due to space constraints, the introduction to $\mathcal{L}$ will be rather succinct, and hence possibly a bit harsh for readers unfamiliar with $\overline{\lambda}\mu\tilde{\mu}$ and / or CBPV. Other articles that could give some intuition are [19, 3, 10, 2, 4, 14, 12, 17, 22, 16, 18, 23, 24]. We would recommend [25, 10, 9] to understand the "abstract-machine-like" part of the calculus, [25, 19] to understand CBPV part, and [9, 12]to understand the relationship with proof theory. Note to reviewers: An unpublished report was sent with this submission (in the file report.pdf). It introduces the calculus in a more pedagogical way, and gives explicit translations from / to the $\lambda$-calculi, and from CBPV (which was announced in the original abstract, but is no longer is the current paper for space reasons). Section 3 of that report is *not* worth reading. A (possibly updated) version of this report will eventually be available on the author's website, and this note will be replaced by a link to it.

We now introduce the $\mathcal{L}$ calculus. A computation is represented by a commands $c = \langle t_\varepsilon \mid\mid e_\varepsilon \rangle^\varepsilon$, with the polarity $\varepsilon$ denoting the current evaluation strategy: $+$ for CBV and $-$ for CBN. In a command $\langle t_\varepsilon \mid\mid e_\varepsilon \rangle^\varepsilon$, the term $t_\varepsilon$ represents the $\lambda$-term $M$ that the "abstract machine" is currently trying to reduce, and $e_\varepsilon$ is the remainder of the term, represented as a context $\mathbb{N}$, i.e. a $\lambda$-term with a hole $\Box$. We write $\mathbb{N}\boxed{M}$ for the *non*-capture-avoiding substitution of $\Box$ by $M$ in $\mathbb{N}$, and we say that $\mathbb{N}\boxed{M}$ is the result of plugging the term $M$ in the hole $\Box$ of the context $\mathbb{N}$. The command $\langle t_\varepsilon \mid\mid e_\varepsilon \rangle^\varepsilon$ then represents the term $\mathbb{N}\boxed{\underline{M}}$, where the underlining represents the focus of the abstract machine. The evaluation context $\tilde{\mu}x^\varepsilon.c$ represents $\mathsf{let}\, x = \Box \,\mathsf{in}\, c$, with an evaluation strategy depending on the polarity $\varepsilon$. The term $\mu\star^\varepsilon.c$ represents the result of the computation $c$. Note that the Lafont critical pair is not present in this calculus (because $\mu\star^+.c_1$ is not a $V_+$, and $\tilde{\mu}x^-.c_2$ is not an $S_-$):

$$\begin{array}{ccccc} c_1\,[\tilde{\mu}x^+.c_2/\star^+] & \lhd & \langle \mu\star^+.c_1 \mid\mid \tilde{\mu}x^+.c_2\rangle^+ & \not\rhd & c_2\,[\mu\star^+.c_1/x^+] \\ c_1\,[\tilde{\mu}x^-.c_2/\star^-] & \not\lhd & \langle \mu\star^-.c_1 \mid\mid \tilde{\mu}x^-.c_2\rangle^- & \rhd & c_2\,[\mu\star^-.c_1/x^-] \end{array}$$

Many types can be added to this base calculus: functions, lazy and strict pairs, sums, and more. See for example figure 5 of [25], or figure 1 of [23]. For our purposes, the exact types often do not matter, so we abstract them away: we have positive types $\tau_1^+, \ldots, \tau_n^+$ and negative types $\tau_1^-, \ldots, \tau_{n'}^-$, and for each type, a certain number of associated constructors and a pattern match that maches all possible constructors of this type. Of course, each constructor takes a fixed number of arguments of a fixed shape. For example, the tensor /

$$
\begin{array}{rcl}
\langle V_\varepsilon \,||\, \tilde{\mu} x^\varepsilon . c\rangle^\varepsilon & \rhd_{\tilde{\mu}^\varepsilon} & c\left[V_\varepsilon / x^\varepsilon\right] \\
\langle \mu \alpha^\varepsilon . c \,||\, S_\varepsilon\rangle^\varepsilon & \rhd_{\mu^\varepsilon} & c\left[S_\varepsilon / \alpha^\varepsilon\right] \\
\left\langle \mathfrak{v}_k^\tau\left(\vec{A}\right) \,||\, \tilde{\mu}\left[\mathfrak{v}_1^\tau\left(\vec{a_1}\right).c_1 \mid \ldots \mid \mathfrak{v}_n^\tau\left(\vec{a_n}\right).c_n\right]\right\rangle^+ & \rhd_{\mathfrak{v}_k^\tau} & c_k\left[\vec{A} / \vec{a_k}\right] \\
\left\langle \mu\langle \mathfrak{s}_1^\tau\left(\vec{a_1}\right).c_1 \mid \ldots \mid \mathfrak{s}_n^\tau\left(\vec{a_n}\right).c_n\rangle \,||\, \mathfrak{s}_k^\tau\left(\vec{A}\right)\right\rangle^- & \rhd_{\mathfrak{s}_k^\tau} & c_k\left[\vec{A} / \vec{a_k}\right]
\end{array}
$$

**Figure 2** Operational / top-level reduction $\rhd$

$$
\begin{array}{lll}
\text{Arguments:} & \text{Variables:} & \text{Whatsits:} \\
A \;::=\; V_\varepsilon \mid S_\varepsilon & a \;::=\; x^\varepsilon \mid \alpha^\varepsilon & w \;::=\; t_\varepsilon \mid e_\varepsilon \mid c
\end{array}
$$

**Figure 3** Notations

strict pair type $\otimes$ has a unique constructor that takes two positive values $\mathfrak{v}_1^\otimes(V_+, W_+)$. The downshift $\Downarrow$ type has a single constructor that takes a negative value $\mathfrak{v}_1^\Downarrow(V_-)$. Often, we will handle constructors quite uniformly, and will therefore write $\mathfrak{v}_k^\tau\left(\vec{A}\right)$ for both.

$$
\begin{array}{lcl}
\text{Positive values:} & & \\
V_+ & ::= & x^+ \mid \mathfrak{v}_1^{\tau_1^+}\left(\vec{A}\right) \mid \ldots \mid \mathfrak{v}_{n_1}^{\tau_1^+}\left(\vec{A}\right) \mid \ldots \\
\text{Positive stacks and evaluation contexts:} & & \\
S_+, e_+ & ::= & \alpha^+ \mid \tilde{\mu} x^+ . c \mid \tilde{\mu}\left[\mathfrak{v}_1^{\tau_1^+}\left(\vec{a_1}\right).c_1 \mid \ldots \mid \mathfrak{v}_{n_1}^{\tau_1^+}\left(\overrightarrow{a_{n_1}}\right).c_{n_1}\right] \mid \ldots \\
\text{Positive terms:} & & \\
T_+ & ::= & \mu \alpha^+ . c \\
t_+ & ::= & V_+ \mid T_+ \\
\text{Negative values amd terms:} & & \\
V_-, t_- & ::= & x^- \mid \mu \alpha^- . c \mid \mu\left\langle \mathfrak{s}_1^{\tau_1^-}\left(\vec{a_1}\right).c_1 \mid \ldots \mid \mathfrak{s}_{n_1}^{\tau_1^-}\left(\overrightarrow{a_{n_1}}\right).c_{n_1}\right\rangle \mid \ldots \\
\text{Negative stacks:} & & \\
S_- & ::= & \alpha^- \mid \mathfrak{s}_1^{\tau_1^-}\left(\vec{A}\right) \mid \ldots \mid \mathfrak{s}_{n_1}^{\tau_1^-}\left(\vec{A}\right) \mid \ldots \\
\text{Negative evaluation contexts:} & & \\
E_- & ::= & \tilde{\mu} x^- . c \\
e_- & ::= & S_- \mid E_- \\
\text{Commands:} & & \\
\mathfrak{c} \ni c & ::= & \langle t_+ \,||\, S_+\rangle^+ \mid \langle V_- \,||\, e_-\rangle^-
\end{array}
$$

**Figure 1** Syntax of $\mathcal{L}$

figure 1 describes the syntax of $\mathcal{L}$, figure 2 describes the top-level reduction $\rhd$ (which corresponds to the weak head reduction of the $\lambda$-calculus), and figure 3 describes notations that we will use to factor statements / proofs. The substitution is defined as expected. We work up to $\alpha$-renaming, always assuming that bound variable are distinct from free variables, and that all the substitutions we manipulate are idempotent.

▶ **Lemma 1.1.** *The top-level reduction $\rhd$ is deterministic: If $c_l \lhd c \rhd c_r$ then $c_l = c_r$.*

**Proof.** Immediate. ◀

▶ **Lemma 1.2.** *The top-level reduction $\triangleright$ is substitutive: For all command $c$ and $c'$, and substitution $\varphi$, if $c \triangleright c'$ then $c\,[\varphi] \triangleright c'\,[\varphi]$.*

A *multicontext* is a whatsit with holes $\square$. A *context* is a multicontext with a single hole. The operation of filling the holes of a context is written $\mathbb{w}\boxed{w_1, \ldots, w_n}$ and when writing this we always assume that the number of whatsits given correspond exactly to the number of hole in the multicontext, and that $\mathbb{w}\boxed{w_1, \ldots, w_n}$ is a whatsit (so that we would never write, for example $(\tilde{\mu}x^+.\square)\boxed{V_+}$ because the hole is at the position of a command, and plugging a value is therefore meaningless). The strong reduction $\to$ is by: $\mathbb{w}\boxed{c} \to \mathbb{w}\boxed{c'}$ whenever $c \triangleright c'$. In other words, $\to$ is the closure under contexts of $\triangleright$.

▶ **Lemma 1.3.** *The strong reduction $\to$ is substitutive: For all command $c$ and $c'$, and substitution $\varphi$, if $c \to c'$ then $c\,[\varphi] \to c'\,[\varphi]$.*

**Proof.** By induction on the syntax. ◀

In the intuitionistic calculus, we want to ensure that no stack is ever discarded or duplicated, i.e. that all stack variables are used linearly. Note that in the presence of additives, one use per branch counts as linear: In $\tilde{\mu}\,[\mathsf{true}\,.\langle x^\varepsilon \mid\mid \star^\varepsilon\rangle^\varepsilon \mid \mathsf{false}\,.\langle y^\varepsilon \mid\mid \star^\varepsilon\rangle^\varepsilon]$, $\star^\varepsilon$ is linearly free, but neither $x^\varepsilon$ nor $y^\varepsilon$ is. Defining "$a$ is linearly free in $w$" directly would involve a lot of case analysis (for example, being linear in $\langle t_\varepsilon \mid\mid e_\varepsilon\rangle^\varepsilon$ means being linear in either one, and not free in the other. We therefore define a more general measure $\lfloor w \rfloor_a$ which is the set of all natural numbers $n$ such that keeping exactly one branch per pattern match leads to a whatsit with $n$ free occurrences of $a$. The addition used in the definition is the pointwise addition of sets, i.e. $\lfloor t_\varepsilon \rfloor_a + \lfloor e_\varepsilon \rfloor_a = \{n_1 + n_2 : n_1 \in \lfloor t_\varepsilon \rfloor_a \wedge n_2 \in \lfloor e_\varepsilon \rfloor_a\}$.

▶ **Definition 1.4.**

$$\lfloor \langle t_\varepsilon \mid\mid e_\varepsilon\rangle^\varepsilon \rfloor_a = \lfloor t_\varepsilon \rfloor_a + \lfloor e_\varepsilon \rfloor_a \qquad \lfloor \mu\alpha^\varepsilon.c \rfloor_a = \lfloor \tilde{\mu}x^\varepsilon.c \rfloor_a = \lfloor c \rfloor_a$$

$$\lfloor \mathfrak{v}_k^\tau\,(A_1, \ldots, A_n) \rfloor_a = \lfloor \mathfrak{s}_k^\tau\,(A_1, \ldots, A_n) \rfloor_a = \lfloor A_1 \rfloor_a + \cdots + \lfloor A_n \rfloor_a$$

$$\lfloor \tilde{\mu}\,[\mathfrak{v}_1^\tau\,(\overrightarrow{a_1})\,.c_1 \mid \ldots \mid \mathfrak{v}_n^\tau\,(\overrightarrow{a_n})\,.c_n] \rfloor_a = \lfloor \mu\langle \mathfrak{s}_1^\tau\,(\overrightarrow{a_1})\,.c_1 \mid \ldots \mid \mathfrak{s}_n^\tau\,(\overrightarrow{a_n})\,.c_n\rangle \rfloor_a = \lfloor c_1 \rfloor_a \cup \cdots \cup \lfloor c_n \rfloor_a$$

We can then define being linearly free in $a$ very easily: A variable $a$ is said to be *linearly free* in $w$ when $\lfloor w \rfloor_a \subseteq \{1\}$. A value constructor $\mathfrak{v}_k^\tau$ is said to be *intuitionistic* if all its arguments are values, and a stack constructor $\mathfrak{s}_k^\tau$ is said to be *intuitionistic* when exactly one of its argument is a stack (and without loss of generality, we will assume that the stack argument is the last one). We say that a whatsit is said to be *intuitionistic* when it only contains intuitionistic constructor, and all its subwhatsits have at most one free stack variable and if it does have one then it is linearly free. From a proof theory perspective, this corresponds to the intuitionistic sequent calculus being the restriction of the classical sequent calculus to sequents having at most one conclusion. An induction on the syntax shows that an intuitionistic term has no free stack variable, and an intuitionistic command / evaluation context has exactly one free stack variable and it is linearly free. This stack variable is often named $\star^\varepsilon$ instead of $\alpha^\varepsilon$ to denote that we are in the intuitionistic fragment. Also note that the restriction to the intuitionistic calculus is very syntactical: the syntax of the intuitionistic fragment is context-free[1].

[1] One just has to split $c$ into $c_{\star^+}$ and $c_{\star^-}$ (and similarly for all syntactic categories of contexts) to keep track of whether the current stack variable is positive or negative.

▶ **Definition 1.5.** A command $c$ is called *normal* if $c \not\triangleright$ and *reducible* otherwise.

▶ **Definition 1.6.** A command $c$ is said to be:

- *Diverging* when $c \triangleright^{\omega}$;
- *Converging* (to $c'$) when $c \triangleright^{*} c' \not\triangleright$;
  - *Clashing* or a *clash* when for all $\varphi$, $c'[\varphi] \not\triangleright$;
  - *Solved* when $c' = \langle x^{\varepsilon} \mid\mid \alpha^{\varepsilon} \rangle^{\varepsilon}$;
  - *Waiting* otherwise (i.e. there exists $\varphi$ such that $c'[\varphi] \triangleright$, but $c' \neq \langle x^{\varepsilon} \mid\mid \alpha^{\varepsilon} \rangle^{\varepsilon}$).

Note that a command is either converging or diverging (by 1.1). Furthermore, a converging command is eventually clashing, eventually solved or eventually waiting.

▶ **Definition 1.7.** We write $\mathcal{L}_{\mathbf{c}}$ for the full classical calculus, $\mathcal{L}_{\mathfrak{ivs}}$ for an intuitionistic fragment with at most one positive value constructor and at most one negative stack constructor, and $\mathcal{L}_{\mathbf{i}\not{\mathfrak{v}}}$ for the calculus in which none of the $V_i$ in $\mathfrak{s}_k^{\tau}\left(\vec{V}, S_{\varepsilon}\right)$ contains a negative value.

We will operationally characterize solvability in $\mathcal{L}_{\mathbf{c}}$, $\mathcal{L}_{\mathfrak{ivs}}$, and $\mathcal{L}_{\mathbf{i}\not{\mathfrak{v}}}$. In the other fragments, we will still have a reduction that is weakly-normalizing exactly on solvable commands, but it may not be decidable.

# 2 Polarized solvability

## 2.1 Definitions

We now define solvability in our calculus. The most common definition in the $\lambda$-calculus is that there exists a substitution $\varphi$ and an applicative context $\Box N_1 \ldots N_m$ such that $(\Box N_1 \ldots N_m)\boxed{M[\sigma]} = M[\sigma] N_1 \ldots N_m \to^{*} I$. In our calculus, the subsitution and the applicative context become a single substition (acting on both value variable and stack variables). To make things not symmetric, we replace $I$ with $x$.

▶ **Definition 2.1.** A substitution $\varphi$ is said to *solve* $c$, written $\varphi \models c$, when $c[\varphi] \to^{*} \langle x^{\varepsilon} \mid\mid \alpha^{\varepsilon} \rangle^{\varepsilon}$.
A command $c$ is called *solvable*, written $\exists \models c$,if there exists a substitution that solves it.

Note that diverging and clashing commands are unsolvable, that solved commands are solvable. Waiting commands however can be either solvable or unsolvable. Solvable commands are either solved or waiting.

In our proof that the ahead reduction operationally characterizes solvability, we will sometimes need to use other reductions in the definition of solvability, hence the following definition.

▶ **Definition 2.2.** A command $c$ is $\rightsquigarrow$-solvable if there exists $\varphi$ such that $c[\varphi] \rightsquigarrow^{*} \langle x^{\varepsilon} \mid\mid \alpha^{\varepsilon} \rangle^{\varepsilon}$.

The proof that the ahead reduction $\rightharpoonup$ operationally characterizes solvability will be done in two steps: The first step, lemma .2, which is described in figure 5, states that $\rightharpoonup$-solvability is equivalent to solvability. The second, which is described in section 2.2, states that $\rightharpoonup$ operationally characterizes $\rightharpoonup$-solvability.

▶ **Theorem 2.3.** *For any reduction $\rightharpoonup$ such that $\triangleright \subseteq \rightharpoonup \subseteq \to$, $\rightharpoonup$-solvability is equivalent to solvability.*

**Proof.** In the appendix. ◀

## 2.2 Operational characterization of solvability

▶ **Definition 2.4.** Given a set $X \subseteq \mathbb{c}$ of commands, we say that a reduction $\rightsquigarrow \subseteq \rightarrow^*$:

- Is $X$-sound when $c \rightsquigarrow^* \not\rightsquigarrow$ implies $c \in X$;
- Is $X$-complete when $c \in X$ implies $c \rightsquigarrow^* \not\rightsquigarrow$;
- Operationally characterizes $X$ when it is $X$-sound, $X$-complete and decidable.

The main theorem of this paper, theorem 6.1, is the existence of $\rightharpoonup \subseteq \rightarrow^*$ such that $\rightharpoonup$ operationally characterizes solvability (i.e. operationally characterizes solvable commands) in some of the $\mathcal{L}$ calculi. Note that if we required only two of solvability-sound, solvability-complete and decidable, then it would be very easy: $\rightarrow$ is solvability-complete and decidable but not solvability-sound, $\emptyset$ is solvability-sound and decidable but not solvability-complete, and the relation $\rightarrow_{\text{unsol}}$, defined by $c \rightarrow_{\text{unsol}} c'$ if and only $c = c'$ and $c$ is unsolvable, is solvability-sound and solvability-complete but not decidable.

Note that while in pure call-by-name and call-by-value $\lambda$-calculus, we can characterize solvability with a reduction $\rightharpoonup \subseteq \rightarrow$, this is no longer possible in more general calculi. In the presence of clashes, since there are $\rightarrow$-normal clashes (for example, $\mathsf{if}\, \lambda x.x \,\mathsf{then}\, y \,\mathsf{else}\, z$), we must as least weaken the inclusion to $\rightharpoonup \subseteq \rightarrow^=$. In the presence of additives, if we want the reduction to be somewhat "regular", i.e. defined as some sort of closure under contexts, we need to reduce in several branches in parallel[2] (for example $\mathsf{if}\, x \,\mathsf{then}\, M_1 \,\mathsf{else}\, M_2 \rightharpoonup \mathsf{if}\, x \,\mathsf{then}\, M_1' \,\mathsf{else}\, M_2'$ whenever $M_1 \rightharpoonup M_1'$ and $M_2 \rightharpoonup M_2'$) so that we have to weaken the inclusion to $\rightharpoonup \subseteq \rightarrow^+$ (where $\rightarrow^+$ could be replaced by the parallel reduction).

In this section we give a generic proof that reduces proving that a reduction $\rightharpoonup$ operationally characterizes $\rightharpoonup$-solvability to proving 3 simpler properties: substitutivity of $\rightharpoonup$, $\rightharpoonup$-solvability of $\rightharpoonup$-normal forms, and $\rightharpoonup$ having uniqueness of termination behavior. The proof is more or less a reformulation of the one given for the call-by-value $\lambda$-calculus in [5], with the slight differences that the diamond property has been weakened to uniqueness of termination behavior, and that we decomposed the proof that $\rightharpoonup$ operationally characterizes solvability in two parts: The first part, given in section 2.1, shows that $\rightharpoonup$-solvability is equivalent to solvability, and the second part, described in this section, shows that $\rightharpoonup$ operationally characterizes $\rightharpoonup$-solvability.

▶ **Definition 2.5** (Uniqueness of termination behavior)**.** A reduction $\rightsquigarrow$ is said to have *uniqueness of termination behavior* if weakly $\rightsquigarrow$-normalizing implies strongly $\rightsquigarrow$-normalizing.

▶ **Lemma 2.6.** *For any reduction $\rightharpoonup \subseteq \rightarrow^*$, if:*

- *(Subst) $\rightharpoonup$ is substitutive;*
- *(NFSol) $\rightharpoonup$-normal implies $\rightharpoonup$-solvable;*
- *(UTB) $\rightharpoonup$ has uniqueness of termination behavior;*

*then $\rightharpoonup$ operationally characterizes $\rightharpoonup$-solvability.*

**Proof.** In the appendix. ◀

#### 2.2.0.1 Pure call-by-name $\lambda$-calculus

In the pure call-by-name $\lambda$-calculus, there are two possible choices for $\rightharpoonup$. The usual one is to take $\rightharpoonup$ equal to the head reduction, i.e. the reduction reducing under contexts of the shape $\lambda x_1. \ldots .\lambda x_n.\square N_1 \ldots N_m$. In this case, (UTB) is trivial because the head

[2] This will be more thoroughly explained in section §3

reduction is deterministic, and (NFSol) is easy because normal forms are of the shape $\lambda x_1.\ldots.\lambda x_n.yN_1\ldots N_m$ so that plugging one in the context $\Box z_1\ldots z_n$ yields a term reducible to a term of the shape $y'N'_1\ldots N'_m$ which is solvable by $[\lambda y_1.\ldots.\lambda y_m.I/y']$. The other choice, closer to our ahead reduction, is to allow reducing under an arbitrary composition of contexts of the shape $\lambda x.\Box$ and $\Box N$, which, in addition to the contexts defining the head reduction, also allow reducing under redexes, for example $(\lambda x.\Box)\,N$. For this alternative reduction, (UTB) is proven by proving the diamond property, and since it has the same normal forms as the head reduction, the proof of (NFSol) does not change.

#### 2.2.0.2 Pure call-by-value $\lambda$-calculus

In the pure call-by-value $\lambda$-calculus, things are more complicated because one has to evaluate arguments before discarding them. In fact, in the $\lambda$-calculus with $\beta$-reduction restricted to values, $(\lambda x.\delta)\,(yz)\,\delta$ is normal and yet unsolvable (because if $yz$ reduces to a value, the whole term reduces to $\Omega = \delta\delta$). Several modifications of the pure call-by-value $\lambda$-calculus were proposed to fix this problem, some of which are described and shown equivalent in [4]. Among those calculi, two are of particular interest to us: $\lambda_{\text{vsub}}$ which is used to operationally characterize solvability in [5] with a proof that we generalize in this paper, and $\lambda_{\text{vseq}}$ which is very similar to our calculus (because both are related to the $\overline{\lambda}\mu\tilde{\mu}$ of [9]). The idea of the $\lambda_{\text{vsub}}$ calculus is to introduce let expressions, and to make them commute with applications. For example:

$$(\lambda x.\delta)\,(yz)\,\delta \to_\beta (\mathsf{let}\,x = yz\,\mathsf{in}\,\delta)\,\delta \to_{\text{com}} \mathsf{let}\,x = yz\,\mathsf{in}\,\delta\delta \to \mathsf{let}\,x = yz\,\mathsf{in}\,\delta\delta$$

The thing that makes $\lambda_{\text{vseq}}$ work is that instead of only having a syntax of terms, it has a syntax of terms (which are represented by commands) and a syntax of values. The importance of this distinction between terms that represent computations and values is explained in [19], where a fine-grained call-by-value $\lambda$-calculus (“partially based on [21]”) is introduced. Very roughly, the idea is that in applications, both the function and the argument have to be values, and to represent $MN$, we therefore either use $\mathsf{let}\,x = M\,\mathsf{in}\,\mathsf{let}\,y = N\,\mathsf{in}\,xy$ or $\mathsf{let}\,y = N\,\mathsf{in}\,\mathsf{let}\,x = M\,\mathsf{in}\,xy$, so that the arbitrary choice in evaluation order is made explicit in the syntax. Through this transformation, $(\lambda x.\delta)\,(yz)\,\delta$ is compiled to $\mathsf{let}\,f = \lambda x.\delta\,\mathsf{in}\,\mathsf{let}\,a = yz\,\mathsf{in}\,\mathsf{let}\,g = fa\,\mathsf{in}\,g\delta$ which diverges as expected:

$$\begin{array}{ll} & \mathsf{let}\,f = \lambda x.\delta\,\mathsf{in}\,\mathsf{let}\,a = yz\,\mathsf{in}\,\mathsf{let}\,g = fa\,\mathsf{in}\,g\delta \\ \to & \mathsf{let}\,a = yz\,\mathsf{in}\,\mathsf{let}\,g = (\lambda x.\delta)\,a\,\mathsf{in}\,g\delta \\ \to & \mathsf{let}\,a = yz\,\mathsf{in}\,\delta\delta \\ \to & \mathsf{let}\,a = yz\,\mathsf{in}\,\delta\delta \end{array}$$

# 3 The ahead reduction

## 3.1 Intuition

Our intuition for defining the ahead reduction in the general case is the following: Since we want the reduction to be substitutive, we want our reduction to handle $x^+$ and an arbitrary value $V_+$ in the same way. The two other properties that we need that but are hard to obtain are (UTB) uniqueness of termination behaviour and (NFSol) solvability of $\rightharpoonup$-normal commands. The next few paragraphs give intuition on how to avoid breaking those two properties.

### Reducing the side that "has the control"

Redexes are due to the interaction of a context with a term, with one of them "having the control" and deciding what happens next, which the other one being somewhat "passive" and gets moved around with no control over its fate. For example, in $(\lambda x.t)\,u$ is the term $\lambda x.t$ "has the control" and the context $\Box u$ is "passive": $\lambda x.t$ moves the $u$ around, and what happens depends heavily on what $t$ is but not at all on what $V$ is. Similarly, in $\mathsf{if}\,t\,\mathsf{then}\,u_1\,\mathsf{else}\,u_2$, the context $\mathsf{if}\,\Box\,\mathsf{then}\,u_1\,\mathsf{else}\,u_2$ "has the control", while the term $t$ is "passive". Another example is $\mathsf{let}\,x = t\,\mathsf{in}\,u$ where $\mathsf{let}\,x = \Box\,\mathsf{in}\,u$ "has the control" and $t$ is passive. In other to ensure uniqueness of termination behavior, we restrict the ahead reduction so that it only reduces whoever "has the control", because otherwise, reducing the "passive" part could lead to divergence, while the part that "has the control" could discard the "passive" part when activated, leading to convergence. An example of this is: $I \lhd (\lambda x.I)\,\Omega \rightharpoonup (\lambda x.I)\,\Omega \rightharpoonup \ldots$. Because we are in the intuitionistic case, reducing the $t$ in $\mathsf{if}\,t\,\mathsf{then}\,u_1\,\mathsf{else}\,u_2$ does not break UTB, even though the $t$ does not "have the control". This is because $t$ can not discard $\mathsf{if}\,\Box\,\mathsf{then}\,u_1\,\mathsf{else}\,u_2$. In the classical setting, $t$ could be a $\mu\alpha.c$ and we would have $\mathsf{if}\,\mu\alpha.c\,\mathsf{then}\,u_1\,\mathsf{else}\,u_2 \to c\,[\mathsf{if}\,\Box\,\mathsf{then}\,u_1\,\mathsf{else}\,u_2/\alpha]$, potentially discarding $\mathsf{if}\,\Box\,\mathsf{then}\,u_1\,\mathsf{else}\,u_2$, and breaking UTB: $I \lhd \mathsf{if}\,\mu\alpha.I\,\mathsf{then}\,\Omega\,\mathsf{else}\,\Omega \rightharpoonup \mathsf{if}\,\mu\alpha.I\,\mathsf{then}\,\Omega\,\mathsf{else}\,\Omega \rightharpoonup \ldots$.

In the $\mathcal{L}$ calculus, in any command, just by looking at the syntactic category of each side of a command, it is possible to know which side "has the control", i.e. contains the subcommand that could get to the top-level after a $\rhd$ reduction step. It is $T_+$ in $\langle T_+ \,||\, S_+\rangle^+$ (because the only possible reduction is $\rhd_{\mu^+}$), $E_-$ in $\langle V_- \,||\, E_-\rangle^-$ (because the only possible reduction is $\rhd_{\tilde\mu^-}$), $S_+$ in $\langle V_+ \,||\, S_+\rangle^+$ (because the only possible reductions are $\rhd_{\tilde\mu^+}$ and $\rhd_{\mathfrak{v}_k^\tau}$), and $V_-$ in $\langle V_- \,||\, S_-\rangle^-$ (because the only possible reductions are $\rhd_{\mu^-}$ and $\rhd_{\mathfrak{s}_k^\tau}$). This corresponds to $\langle \mathbb{T}_+ \,||\, S_+\rangle^+$, $\langle V_+ \,||\, \mathbb{S}_+\rangle^+$, $\langle V_- \,||\, \mathbb{E}_-\rangle^-$ and $\langle \mathbb{V}_- \,||\, S_-\rangle^-$ being ahead contexts. In the intuitionistic calculus, since stack variables are always used linearly, the synchronized diamond property will not be broken by reducing the $S_+$ in $\langle T_+ \,||\, S_+\rangle^+$, or the $S_-$ in $\langle V_- \,||\, S_-\rangle^-$. This corresponds to $\langle T_+ \,||\, \mathbb{S}_+\rangle^+$ and $\langle V_- \,||\, \mathbb{S}_-\rangle^-$ being ahead contexts.

### Reducing in parallel

In the presence of additives (e.g. booleans or negative / lazy pairs), the ahead reduction has to reduce in each branch in parallel. There are two reasons for this. The first reason is that an if-then-else $\mathsf{if}\,x\,\mathsf{then}\,t_1\,\mathsf{else}\,t_2$(where $x$ is free neither in $t_1$ nor in $t_2$) is solvable whenever $t_1$ is (because we can take $[\mathsf{true}/x]$) or $t_2$ is (because we can take $[\mathsf{false}/x]$), and only in those two cases (because if we pick any other value for $x$, the result is a clash, which is not solvable). Ensuring that $\mathsf{if}\,x\,\mathsf{then}\,t_1\,\mathsf{else}\,t_2 \rightharpoonup \mathsf{if}\,x\,\mathsf{then}\,t_1'\,\mathsf{else}\,t_2'$ whenever $t_1 \rightharpoonup t_1'$ and $t_2 \rightharpoonup t_2'$ ensures this. The second reason is that always allowing to reduce only on one side would break the synchronized diamond property. For example, if we allow reducing only in the first term, by substitutivity we would get the peak $t_2 \lhd \mathsf{if}\,\mathsf{false}\,\mathsf{then}\,t_1\,\mathsf{else}\,t_2 \rightharpoonup \mathsf{if}\,\mathsf{false}\,\mathsf{then}\,t_1'\,\mathsf{else}\,t_2$ and there would in general be now way to close this peak: one has $\mathsf{if}\,\mathsf{false}\,\mathsf{then}\,t_1'\,\mathsf{else}\,t_2 \rhd t_2$ but in general we do not have $t_2 \rightharpoonup t_2$. For the exact same reasons, we should have $(M, N) \rightharpoonup (M', N')$ whenever $M \rightharpoonup M'$ and $N \rightharpoonup N'$ (with the slight difference that now substitutivity is substitutivity with respect to stack variables, which allows to deduce $\pi_i\,(M, N) \rightharpoonup \pi_i\,(M', N')$ from $(M, N) \rightharpoonup (M', N')$). In the $\mathcal{L}$ calculus, the same thing happens, and the syntax makes the symmetry clearer: $\mathsf{if}\,\Box\,\mathsf{then}\,t_1\,\mathsf{else}\,t_2$ becomes $\tilde\mu\,[\mathsf{true}\,.c_1 \mid \mathsf{false}\,.c_2]$, and $(M_1, M_2)$ becomes $\mu\langle\,(\pi_1 \cdot \star^-)\,.c_1 \mid (\pi_2 \cdot \star^-)\,.c_2\rangle$.

### Detecting dead branches

Another difficulty that arises when adding additives is that some branches are clearly inaccessible / dead, but not $\to$ reduction step can erase them. For example, $\mathsf{if}\,x\,\mathsf{then}\,(\mathsf{if}\,x\,\mathsf{then}\,\Omega\,\mathsf{else}\,I)\,\mathsf{else}\,\Omega$ is not solvable: both $[\mathsf{true}/x]$ and $[\mathsf{false}/x]$ lead to $\Omega$, and any other substitution either does nothing or leads to a crash. This should be contrasted with $\mathsf{if}\, x\, \mathsf{then}\, (\mathsf{if}\, y\, \mathsf{then}\, \Omega\, \mathsf{else}\, I)\, \mathsf{else}\, \Omega$ which is solved by $[\mathsf{true}/x, \mathsf{false}/y]$. Unfortunately, if the ahead reduction reduces in parallel in all branches, then $\mathsf{if}\, x\, \mathsf{then}\, (\mathsf{if}\, x\, \mathsf{then}\, \Omega\, \mathsf{else}\, I)\, \mathsf{else}\, \Omega$ will be ahead normal, hence breaking solvability of normal forms.

One way to solve this problem would be to reason modulo some relation $\sim \subsetneq =_{\beta\eta}$ which will use $\eta$ rules to propagate information. Indeed, writing $\mathbb{K}$ for $\mathsf{let}\, y = \square\, \mathsf{in}\, \mathsf{if}\, y\, \mathsf{then}\, (\mathsf{if}\, y\, \mathsf{then}\, \Omega\, \mathsf{else}\, I)\, \mathsf{else}\, \Omega$, we would have

$$\mathsf{if}\, x\, \mathsf{then}\, (\mathsf{if}\, x\, \mathsf{then}\, \Omega\, \mathsf{else}\, I)\, \mathsf{else}\, \Omega \lhd \mathbb{K}\boxed{x} =_\eta \mathsf{if}\, x\, \mathsf{then}\, \mathbb{K}\boxed{\mathsf{true}}\, \mathsf{else}\, \mathbb{K}\boxed{\mathsf{false}}$$

$$\mathsf{if}\, x\, \mathsf{then}\, \mathbb{K}\boxed{\mathsf{true}}\, \mathsf{else}\, \mathbb{K}\boxed{\mathsf{false}} \rightharpoonup \mathsf{if}\, x\, \mathsf{then}\, (\mathsf{if}\, \mathsf{true}\, \mathsf{then}\, \Omega\, \mathsf{else}\, I)\, \mathsf{else}\, \Omega \rightharpoonup \mathsf{if}\, x\, \mathsf{then}\, \Omega\, \mathsf{else}\, \Omega$$

More generally, we would $\sim$ to identify $\mathsf{if}\, x\, \mathsf{then}\, t_1\, \mathsf{else}\, t_2$ and $\mathsf{if}\, x\, \mathsf{then}\, t_1\, [\mathsf{true}/x]\, \mathsf{else}\, t_2\, [\mathsf{false}/x]$. This approach would be slightly unsatisfying because we would no longer have $\rightharpoonup \subsetneq \to^*$ (whereas the approach we will describe later will preserve this inclusion), and very hard to work with because it substitutes free variables. This makes it very hard to reason locally (because free variables could appear elsewhere in the term). In order for $\rightharpoonup$ to be substitutive, the $\sim$ equivalence has to be pretty complex. Indeed, since

$$\begin{array}{l}\mathsf{match}\, x\, \mathsf{with}\, \iota_1\, (y_1) \rightarrowtail I \mid \iota_2\, (y_2) \rightarrowtail \\ \quad \mathsf{match}\, x\, \mathsf{with}\, \iota_1\, (z_1) \rightarrowtail I \mid \iota_2\, (z_2) \rightarrowtail I\end{array} =_\eta \begin{array}{l}\mathsf{match}\, x\, \mathsf{with}\, \iota_1\, (y_1) \rightarrowtail I \mid \iota_2\, (y_2) \rightarrowtail \\ \quad \mathsf{match}\, \iota_2\, (y_2)\, \mathsf{with}\, \iota_1\, (z_1) \rightarrowtail I \mid \iota_2\, (z_2) \rightarrowtail I\end{array}$$

by applying the substitution $[\iota_2\, (V)/x]$, we would expect

$$\begin{array}{l}\mathsf{match}\, \iota_2\, (V)\, \mathsf{with}\, \iota_1\, (y_1) \rightarrowtail I \mid \iota_2\, (y_2) \rightarrowtail \\ \quad \mathsf{match}\, \iota_2\, (V)\, \mathsf{with}\, \iota_1\, (z_1) \rightarrowtail I \mid \iota_2\, (z_2) \rightarrowtail I\end{array} \sim \begin{array}{l}\mathsf{match}\, \iota_2\, (V)\, \mathsf{with}\, \iota_1\, (y_1) \rightarrowtail I \mid \iota_2\, (y_2) \rightarrowtail \\ \quad \mathsf{match}\, \iota_2\, (y_2)\, \mathsf{with}\, \iota_1\, (z_1) \rightarrowtail I \mid \iota_2\, (z_2) \rightarrowtail I\end{array}$$

but this is no longer true by just $=_\eta$. This would require the more general

$$\mathsf{match}\, V\, \mathsf{with}\, \iota_1\, (y_1) \rightarrowtail t_1 \mid \iota_2\, (y_2) \rightarrowtail t_2 \sim \mathsf{match}\, V\, \mathsf{with}\, \iota_1\, (y_1) \rightarrowtail t_1\, [V_1/x] \mid \iota_2\, (y_2) \rightarrowtail t_2\, [V_2/x]$$

whenever any substitution $\varphi$ that unifies $V$ and $\iota_i\, (y_i)$ also unifies $V$ and $V_i$. It might be possible to make this approach work but we found proving UTB with it challenging, and therefore decided to use another approach.

The other approach is that instead of having the reduction propagate the information "$V$ was matched against $\iota_1\, (y)$ somewhere above", we keep this information in the reduction. To do this, we record the context under which we are reducing above the reduction: $\overset{\mathbb{K}}{\rightharpoonup}$. For example,

$$\mathsf{match}\, V\, \mathsf{with}\, \iota_1\, (y_1) \rightarrowtail t_1 \mid \iota_2\, (y_2) \rightarrowtail t_2 \overset{\mathbb{K}}{\rightharpoonup} \mathsf{match}\, V\, \mathsf{with}\, \iota_1\, (y_1) \rightarrowtail t_1' \mid \iota_2\, (y_2) \rightarrowtail t_2'$$

whenever

$$t_1 \overset{\mathbb{K}\boxed{\mathsf{match}\, V\, \mathsf{with}\, \iota_1(y_1) \rightarrowtail \square \mid \iota_2(y_2) \rightarrowtail t_2}}{\rightharpoonup} t_1' \text{ and } t_1 \overset{\mathbb{K}\boxed{\mathsf{match}\, V\, \mathsf{with}\, \iota_1(y_1) \rightarrowtail t_1 \mid \iota_2(y_2) \rightarrowtail \square}}{\rightharpoonup} t_1'$$

We can then allow (notice that it is $t_2$ on both side, there is no $t_2'$)

$$\mathsf{match}\, V\, \mathsf{with}\, \iota_1\, (y_1) \rightarrowtail t_1 \mid \iota_2\, (y_2) \rightarrowtail t_2 \overset{\mathbb{K}}{\rightharpoonup} \mathsf{match}\, V\, \mathsf{with}\, \iota_1\, (y_1) \rightarrowtail t_1' \mid \iota_2\, (y_2) \rightarrowtail t_2$$

if $\mathbb{K}$ is of the shape $\mathbb{K}_1\boxed{\mathsf{match}\, V\, \mathsf{with}\, \iota_1\, (y_1) \rightarrowtail \mathbb{K}_2 \mid \iota_2\, (y_2) \rightarrowtail u_2}$ because we know that in the full term, the $t_2$ branch is dead. This rule would not be enough, as shown by

$$\mathsf{match}\, x\, \mathsf{with}\, \iota_1\, (y_1) \rightarrowtail \mathsf{if}\, y_1\, \mathsf{then} \left( \begin{array}{l} \mathsf{match}\, x\, \mathsf{with} \\ \mid \iota_1\, (z_1) \rightarrowtail \mathsf{if}\, z_1\, \mathsf{then}\, \Omega\, \mathsf{else}\, I \\ \mid \iota_2\, (z_2) \rightarrowtail I \end{array} \right) \mathsf{else}\, \Omega \mid \iota_2\, (y_2) \rightarrowtail \Omega$$

which might give the impression that it is solvable but is not. The first $\mathsf{match}$ forces $x \sim \iota_1(y_1)$ and the first $\mathsf{ifthenelse}$ forces $y_1 \sim \mathsf{true}$. The reduction we described above would detect that the second branch of the second $\mathsf{match}$ is dead (because $x \sim \iota_1(y_1)$), but would not infer $x \sim \iota_1(\mathsf{true})$ from the two previous equations, and would therefore not detect that the second branch of the second $\mathsf{ifthenelse}$ is dead. Another way to think about this is that the term $\mathsf{if}\, x\, \mathsf{then}\, t\, \mathsf{else}\, \Omega$ is solvable if and only if the term $t$ is solved by a substitution $\varphi$ such that $\varphi(x) = \mathsf{true}$. In other words, the context $\mathsf{if}\, x\, \mathsf{then}\, \Box\, \mathsf{else}\, \Omega$ restricted the set of substitutions that could be used to prove that the term is solvable. We make this formal by saying that $\psi$ is available under $\mathbb{K}$ if there exists some $\varphi$ such that $\mathbb{K}[\varphi] \rhd^* \Box[\psi]$ (i.e. for all $c$, $\mathbb{K}\boxed{c}[\varphi] \rhd^* c[\psi]$). We could actually define $\rhd$ directly on contexts if we were careful enough with how we handle substitutions, for example as described in [13], but for our purposes, taking $\rhd$ on contexts as a notation is sufficient. By restricting detection of dead branches to contexts $\mathbb{K}$ of a specific shape (which is more or less “no redex above the hole”), we are able to prove that this property, and hence the $\rightharpoonup$ reduction which relies on it, are decidable in some interesting versions of the calculus.

The detection of dead branches described above also solves all problems related to branches being dead because of clashes in our calculus. For example, any branch placed in the context $\mathsf{if}\, \iota_1(V)\, \mathsf{then}\, \Box\, \mathsf{else}\, t$ is dead because there is no way to have $\iota_1(V) \sim \mathsf{true}$.

## Detecting forced unsolvability

Sometimes, a command $c$ can not be decomposed as $\mathbb{K}\boxed{c_0}$ such that $\mathbb{K}$ allows to detect the unsolvability, even though $c$ is unsolvable. Those cases happen when $\mathbb{K}$ restricts the available substitutions to only those that will make $c_0$ unsolvable. For example the term $\mathsf{let}\, x = yV\, \mathsf{in}\, \pi_1 y$ will be clashing, but the corresponding command $\langle y^- \mid\mid V \cdot \{\tilde{\mu}x^+.\langle y^- \mid\mid \pi_1 \cdot \star^- \rangle^- \}\, \rangle^-$ can at most be decomposed into $\langle y^- \mid\mid V \cdot \{\tilde{\mu}x^+.\Box\}\, \rangle^-$ and $\langle y^- \mid\mid \pi_1 \cdot \star^- \rangle^-$. We therefore generalize a bit our detection of dead branches: We say that $c$ is solvable under $\mathbb{K}$ if there exists $\varphi$ and $\psi$ such that $\mathbb{K}[\varphi] \rhd^* \Box[\psi]$ and $c[\psi] \rhd^* \langle x^\varepsilon \mid\mid \alpha^\varepsilon \rangle^\varepsilon$. To get decidability of $\rightharpoonup$ in some interesting cases, we restrict the shape of $\mathbb{K}$ as previously, and ask that $c$ is indecomposable: There is no ahead context $\mathbb{K}'$ such that $c = \mathbb{K}'\boxed{c_0}$.

## The remaining obstacle: separability

Out reduction $\rightharpoonup$ will fail only in intuitionistic calculi where separability of stacks of the shape $\mathfrak{s}_k^\tau\left(\vec{V}, \star^\varepsilon\right)$ is undecidable. The problem is exemplified by the term $\mathsf{if}\, xV_1\, \mathsf{then}\, \mathsf{if}\, xV_2\, \mathsf{then}\, \Omega\, \mathsf{else}\, I\, \mathsf{else}\, \Omega$: This term is solvable if and only if $xV_1 \sim \mathsf{true}$ and $xV_2 \sim \mathsf{false}$, which is exactly the definition of $V_1$ and $V_2$ being separable. Note that this would not be a problem in the classical calculus because we would substitute $x$ by a $\mu$ that would just discard everything, and the term would be solvable. In the intuitionistic fragment, our solution for now is to restrict ourselves to subfragments where separability of stacks of the shape $\mathfrak{s}_k^\tau\left(\vec{V}, \star^\varepsilon\right)$ is decidable. The subfragment where $\mathfrak{s}_k^\tau\left(\vec{V}, \star^\varepsilon\right)$ never contains a negative value, so that all the values it contains are hereditarily positive, i.e. made only of $\mathfrak{v}_k^\tau$ constructors, and separability is therefore decidable (by check if $\tau$ and $k$ match or not, and if both do checking subvalues recursively). The other way to ensure that separability is decidable is to have at most one positive constructor, so that no two stacks are separable.

## 3.2 Definition

Positive values:

$$\mathbb{V}_+ \quad ::= \qquad \text{nothing}$$

Positive stacks and evaluation contexts:

$$\mathbb{S}_+, \mathbb{e}_+ \quad ::= \quad \tilde{\mu} x^+ . \mathbb{c} \quad | \quad \tilde{\mu} \left[ \mathfrak{v}_1^{\tau_1^+} \left( \overrightarrow{a_1} \right) . \mathbb{c}_1 \mid \dots \mid \mathfrak{v}_{n_1}^{\tau_1^+} \left( \overrightarrow{a_{n_1}} \right) . \mathbb{c}_{n_1} \right] \quad | \quad \dots$$

Positive terms:

$$\mathbb{T}_+, \mathbb{t}_+ \quad ::= \quad \mu \alpha^+ . \mathbb{c}$$

Negative values and terms:

$$\mathbb{V}_-, \mathbb{t}_- \quad ::= \quad \mu \alpha^- . \mathbb{c} \quad | \quad \mu \left\langle \mathfrak{s}_1^{\tau_1^-} \left( \overrightarrow{a_1} \right) . \mathbb{c}_1 \mid \dots \mid \mathfrak{s}_{n_1}^{\tau_1^-} \left( \overrightarrow{a_{n_1}} \right) . \mathbb{c}_{n_1} \right\rangle \quad | \quad \dots$$

Negative stacks:

$$\mathbb{S}_- \quad ::= \quad \mathfrak{s}_1^{\tau_1^-} \left( \vec{V}, \mathbb{S}_\varepsilon \right) \mid \dots \mid \mathfrak{s}_{n_1}^{\tau_1^-} \left( \vec{V}, \mathbb{S}_\varepsilon \right) \quad | \quad \dots \qquad (\mathbf{i})$$

$$\text{nothing} \qquad (\mathbf{c})$$

Negative evaluation contexts:

$$\mathbb{E}_- \quad ::= \quad \tilde{\mu} x^- . \mathbb{c}$$

$$\mathbb{e}_- \quad ::= \quad \mathbb{E}_- \quad | \quad \mathbb{S}_-$$

Commands:

$$\mathbb{c} \quad ::= \quad \square \quad | \quad \langle \mathbb{T}_+ \,||\, S_+ \rangle^+ \quad | \quad \langle V_+ \,||\, \mathbb{S}_+ \rangle^+ \quad | \quad \langle V_- \,||\, \mathbb{E}_- \rangle^- \quad | \quad \langle \mathbb{V}_- \,||\, S_- \rangle^-$$
$$| \; \langle T_+ \,||\, \mathbb{S} \rangle^+ \mid \langle V_- \,||\, \mathbb{S}_- \rangle^- \quad (\mathbf{i})$$

**Figure 4** $l$-ahead multicontexts

We start by definition the contexts that will allow us to reduce in parallel in the right places.

▶ **Definition 3.1.** A *multicontext* is a term with several holes.

An $l$*-ahead multicontext* (where $l = \mathbf{i}$ for intuitionistic or $l = \mathbf{c}$ for classical) is a multicontext of the shape described in figure 4. The $l$ will sometimes be made implicit.

An $l$-ahead context is the result of plugging all holes of an $l$-ahead multicontext except one.

▶ **Lemma 3.2.** *If* $\mathbb{w}$ *is a* $l$*-ahead multicontext then* $\mathbb{w}[\varphi] = \mathbb{w}'\boxed{\square[\psi_1], \dots, \square[\psi_n]}$ *where* $\mathbb{w}'$ *is a* $l$*-ahead multicontext.*

▶ **Definition 3.3.** A substitution $\varphi$ solves $c$ under the context $\mathbb{K}$, written $\varphi \models (\mathbb{K}, c)$, if there exists $\psi$ such that $\mathbb{K}[\varphi] \rhd^* \square[\psi]$ and $\psi \models c$.

A command $c$ is solvable under a context $\mathbb{K}$, written $\exists \models (\mathbb{K}, c)$, if there exists $\varphi$ such that $\varphi \models (\mathbb{K}, c)$.

We then define bad sets as approximation of "dead branches".

▶ **Definition 3.4.** $\Omega$ is called a *bad set* when:

- (Bad-unsol) For all $(\mathbb{K}, c) \in \Omega$, $c$ is not solvable under $\mathbb{K}$;
- (Bad-subst) If $(\mathbb{K}, c) \in \Omega$, and $\mathbb{K}[\varphi] = \mathbb{K}'\boxed{\square[\psi]}$ then $(\mathbb{K}', c[\psi]) \in \Omega$;
- (Bad-move) If $(\mathbb{K}_1, \mathbb{K}_2\boxed{c}) \in \Omega$ then $(\mathbb{K}_1\boxed{\mathbb{K}_2}, c) \in \Omega$;
- (Bad-red-K) If $(\mathbb{K}, c) \in \Omega$ and $\mathbb{K} \to \mathbb{K}'\boxed{\square[\varphi]}$ then $(\mathbb{K}', c[\varphi]) \in \Omega$;
- (Bad-red-c) If $(\mathbb{K}, c) \in \Omega$ and $c \to c'$ then $(\mathbb{K}, c') \in \Omega$.

The set $\Omega_{\mathrm{sem}} := \{(\mathbb{K}, c) : c \text{ is not solvable under } \mathbb{K}\}$ is an undecidable bad set. We will later construct a decidable bad set for some versions of the calculus. Given a bad set $\Omega$, we can formalize the intuition we gave about dead branches as described below: We reduce under an ahead multicontext, and for each hole, we reduce the command by one step, unless we are in a dead branch. Note that contexts retain more information that needed: Contexts up to commutations $\mathbb{K}_1\boxed{\mathbb{K}_2} \rightsquigarrow \mathbb{K}_2\boxed{\mathbb{K}_1}$ when they do not bind each other's variables, and where branches not above the hole are forgotten, would still have enough information: The reduction $\overset{\mathbb{K}}{\rightharpoonup}$ is the same for $\mathbb{K} = \mathsf{if}\, x\, \mathsf{then}\, \mathsf{if}\, y\, \mathsf{then}\, \square\, \mathsf{else}\, M_1\, \mathsf{else}\, M_2$ and $\mathbb{K} = \mathsf{if}\, y\, \mathsf{then}\, \mathsf{if}\, x\, \mathsf{then}\, \square\, \mathsf{else}\, \Omega\, \mathsf{else}\, \mathrm{Crash}$ (assuming that $\Omega$ is stable under those transformations, but if it is not, we can complete while preserving being a bad set and being decidable).

▶ **Definition 3.5.**

$$\frac{c \triangleright c' \qquad \mathbb{K}\ l\text{-ahead context}}{c \overset{\mathbb{K}}{\blacktriangleright}_{l,\Omega} c'} \qquad\qquad \frac{(\mathbb{K}, c) \in \Omega \qquad \mathbb{K}\ l\text{-ahead context}}{c \overset{\mathbb{K}}{\blacktriangleright}_{l,\Omega} c}$$

$$\frac{c_1 \overset{\mathbb{K}\boxed{\mathbb{w}\boxed{\square, c_2, \ldots, c_n}}}{\blacktriangleright}_{l,\Omega} c_1' \qquad \ldots \qquad c_1 \overset{\mathbb{K}\boxed{\mathbb{w}\boxed{c_1, \ldots, c_{n-1}, \square}}}{\blacktriangleright}_{l,\Omega} c_1'}{\mathbb{w}\boxed{c_1, \ldots, c_n} \overset{\mathbb{K}}{\rightharpoonup}_{l,\Omega} \mathbb{w}\boxed{c_1', \ldots, c_n'}}$$

▶ **Lemma 3.6.** *If* $w \overset{\mathbb{K}}{\blacktriangleright}_{l,\Omega} w'$ *and* $\mathbb{K}[\varphi] = \mathbb{K}'\boxed{\square\,[\psi]}$ *then* $w\,[\psi] \overset{\mathbb{K}'}{\blacktriangleright}_{l,\Omega} w'\,[\psi]$.

**Proof.** By lemma 1.2 and (Bad-subst). ◀

▶ **Lemma 3.7** (Subst)**.** *If* $w \rightharpoonup w'$ *then* $w\,[\varphi] \rightharpoonup w'\,[\varphi]$.

**Proof.** In the appendix. ◀

## 4 Solvable implies strongly ahead normalizing

By lemma 2.6, the only remaining property to prove is uniqueness of termination behavior. In the call-by-name $\lambda$-calculus, the uniqueness of termination behavior is trivial because $\rightharpoonup$ is the head reduction which is deterministic. In the call-by-value $\lambda$-calculus, the proof of UTB given in [5] relies on proving the diamond property: Whenever $M_l \leftharpoonup M \rightharpoonup M_r$, either $M_l = M_r$ or there exists $M'$ such that $M_l \rightharpoonup M' \leftharpoonup M_r$. Unfortunately, this property if false in the presence of additives. For example (where the $\rightharpoonup$ reduction reduces the $\Omega$ in the else branch):

$$\not\rightleftharpoons \mathsf{if}\, x\, \mathsf{then}\, I\, \mathsf{else}\, \Omega \leftharpoonup \mathsf{if}\, x\, \mathsf{then}\, I\,I\, \mathsf{else}\, (\lambda y.\Omega)\, I \rightharpoonup \mathsf{if}\, x\, \mathsf{then}\, I\, \mathsf{else}\, (\lambda y.\Omega)\, I \not\rightleftharpoons$$

The same example would work for any $M_1$ such that $M_1 \rightharpoonup M_1' \not\rightleftharpoons$ and $M_2$ such that $M_{2,l} \leftharpoonup M_2 \rightharpoonup M_{2,r}$ with $M_{2,l} \neq M_{2,r}$, at least when $y$ is not free in $M_1$ (which prevents $M_1 \overset{\mathbb{N}}{\blacktriangleright}$):

$$\not\rightleftharpoons \mathsf{if}\, y\, \mathsf{then}\, M_1'\, \mathsf{else}\, M_{2,l} \leftharpoonup \mathsf{if}\, y\, \mathsf{then}\, M_1\, \mathsf{else}\, M_2 \rightharpoonup \mathsf{if}\, y\, \mathsf{then}\, M_1'\, \mathsf{else}\, M_{2,r} \not\rightleftharpoons$$

The problem is that both branches of the if are synchronized, so that even though the two reductions in the else branch could potentially be joined, they are blocked by the if branch. One could try to weaken the diamond property to: Whenever $M_l \leftharpoonup M \rightharpoonup M_r$, either $M_l$ and $M_r$ are both normal or there exists $M'$ such that $M_l \rightharpoonup M' \leftharpoonup M_r$. However, this still is not enough as show by the following counter-example:

$$\mathsf{if}\, \Omega\, \mathsf{then}\, M_1'\, \mathsf{else}\, M_{2,l} \leftharpoonup \mathsf{if}\, \Omega\, \mathsf{then}\, M_1\, \mathsf{else}\, M_2 \rightharpoonup \mathsf{if}\, \Omega\, \mathsf{then}\, M_1'\, \mathsf{else}\, M_{2,r}$$

The intuition is the same as in the previous counter-example, i.e. the $\mathsf{if}$ branch prevents the $\mathsf{else}$ branch from joining, except that we added redexes above to prevent normalization. Thinking about those two terms a bit more, one can see that any reduction in $\mathsf{if}\,\Omega\,\mathsf{then}\,M_1'\,\mathsf{else}\,M_{2,l}$ corresponds exactly to one reduction in $\mathsf{if}\,\Omega\,\mathsf{then}\,M_1'\,\mathsf{else}\,M_{2,r}$, hence the idea of proving that $\leftharpoondown\cdot\rightharpoonup$, is a bisimulation for $\rightharpoonup$: If $M_l \leftharpoondown\cdot\rightharpoonup M_r \rightharpoonup M_r'$ then there exists $M_l'$ such that $M_l \rightharpoonup M_{l'} \leftharpoondown\cdot\rightharpoonup M_{r'}$. From this, one can prove that if $M_l \leftharpoondown^n\cdot\rightharpoonup^n M_r$ then $M_l\,(\leftharpoondown\cdot\rightharpoonup)^n\,M_r$, and hence that $M_l$ is normal if and only if $M_r$ is. This allows us to conclude that $\rightharpoonup$ has uniqueness of termination behavior.

The next section prove that $\leftharpoondown\cdot\rightharpoonup$ is indeed a bisimulation for $\rightharpoonup$, which involves a lot of case analysis, and is mostly unsurprising, except maybe the presence of lemma 4.1 which gives some intuition on why the intuitionistic case is special. The following section proves that from $\leftharpoondown\cdot\rightharpoonup$ being a bisimulation for $\rightharpoonup$, one can prove that $\rightharpoonup$ has uniqueness of termination behavior, the proof of which is very generic and could apply to other calculi.

▶ **Lemma 4.1.** *In the intuitionistic calculi, if $S_\varepsilon \overset{\mathbb{K}}{\rightharpoonup} S_\varepsilon'$ then $c\,[S_\varepsilon/\star^\varepsilon] \overset{\mathbb{K}}{\rightharpoonup} c\,[S_\varepsilon'/\star^\varepsilon]$.*

**Proof.** By induction the syntax, using the fact that $\star^\varepsilon$ is linearly free. ◀

▶ **Lemma 4.2.** *If $c_l \overset{\mathbb{K}}{\blacktriangleleft} c \left(\overset{\mathbb{K}}{\rightharpoonup} \setminus \overset{\mathbb{K}}{\blacktriangleright}\right) c_r$ then there exists $c'$ such that $c_l \overset{\mathbb{K}}{\rightharpoonup} c' \overset{\mathbb{K}}{\blacktriangleleft} c_r$.*

▶ **Lemma 4.3** (Bisimulation). *$\leftharpoondown\cdot\rightharpoonup$ is a bisimulation for $\rightharpoonup$: If $c_l \leftharpoondown c \rightharpoonup c_r \rightharpoonup c_r'$ then there exists $c_l'$ and $c'$ such that $c_l \rightharpoonup c_l' \leftharpoondown c' \rightharpoonup c_r'$.*

▶ **Lemma 4.4** (Uniqueness of termination behavior). *The ahead reduction has uniqueness of termination behavior: If $c \rightharpoonup^* c' \not\rightharpoonup$ then $c \not\rightharpoonup^\omega$.*

**Proof.** All 3 proofs are in the appendix. ◀

## 5 Decidable ahead reduction

The bad set $\Omega_{\text{sem}}$ is decidable in none of the calculi, and the associated reduction $\rightharpoonup_{\Omega_{\text{sem}}}$ is therefore not decidable. In order to

▶ **Definition 5.1.** An ahead context $\mathbb{K}$ is said to be *reduced* when it is not of the shape $\mathbb{K}_0\boxed{\langle V_\varepsilon \,||\, \tilde{\mu}x^\varepsilon.\mathbb{K}_1\rangle^\varepsilon}$, $\mathbb{K}_0\boxed{\langle \mu\alpha^\varepsilon.\mathbb{K}_1 \,||\, S_\varepsilon\rangle^\varepsilon}$, $\mathbb{K}_0\boxed{\left\langle \mathfrak{v}_k^\tau\left(\vec{A}\right) \,||\, \tilde{\mu}\left[\mathfrak{v}_1^\tau\left(\vec{a_1}\right).\mathbb{K}_1 \mid \ldots \mid \mathfrak{v}_n^\tau\left(\vec{a_n}\right).\mathbb{K}_n\right] \right\rangle^+}$ or $\mathbb{K}_0\boxed{\left\langle \mu\langle\mathfrak{s}_1^\tau\left(\vec{a_1}\right).\mathbb{K}_1 \mid \ldots \mid \mathfrak{s}_n^\tau\left(\vec{a_n}\right).\mathbb{K}_n\rangle \,||\, \mathfrak{s}_k^\tau\left(\vec{A}\right) \right\rangle^-}$.

A command $c$ is said to be *indecomposable* if it is not of the shape $\mathbb{K}\boxed{c_0}$ where $\mathbb{K}$ is an ahead context.

Given this definitions, we can define a bad set that will be decidable in some of the calculi:

▶ **Definition 5.2.** $\Omega_{\text{syn}} := \{(\mathbb{K}, c) \mid \mathbb{K} \text{ reduced} \wedge c \text{ indecomposable } \wedge \not\exists \not\models (\mathbb{K}, c)\}$

The intuition behind why this is decidable is that we removed all unsolvability that was due to non-termination: $\mathbb{K}$ is reduced and therefore has no redex above the hole, and $c$ is indecomposable and $\overset{\mathbb{K}}{\rightharpoonup}_\emptyset$-normal (which, if $\mathbb{K}$ does not have a clash above the hole, is equivalent to $\triangleright$-normal since it is indecomposable). The only remaining obstacles to solvability are therefore clashes and dead branches, and the presence of these obstacles can be decided in well-chosen fragments of the calculus.

We will show that $\Omega_{\text{syn}}$ is decidable in $\mathcal{L}_{\mathbf{c}}$, $\mathcal{L}_{\mathbf{i}\mathfrak{v}\mathfrak{s}}$ and $\mathcal{L}_{\mathbf{i}\not{\mathfrak{v}}}$. The lemma that says that normal forms are solvable is easy because we "cheated" by making $\Omega$, and hence the reduction, speak about solvability, and the difficulty is therefore pushed to the proof that $\Omega$ is decidable.

▶ **Lemma 5.3** (Solvability of normal forms). *If $c \not\rightharpoonup_{\Omega_{syn}}$ then $c$ is solvable.*

**Proof.** Decompose $c$ as $c = \mathbb{c}_0\boxed{c_1, \dots, c_n}$ where for each $k$, $c_k$ is indecomposable. Since $c \not\rightharpoonup_{\Omega_{\text{syn}}}$, we in particular have $c \not\rightharpoonup_{\emptyset}$ so that there exists a $k$ such $\mathbb{K} := \mathbb{c}_0\boxed{c_1, \dots c_{k-1}, \Box, c_{k+1}, c_n}$ is reduced. Since $c \not\rightharpoonup_{\Omega_{\text{syn}}}$, we also have $(\mathbb{K}, c_k) \notin \Omega_{\text{syn}}$. Since $c_k$ is indecomposable and $\mathbb{K}$ is reduced, $(\mathbb{K}, c_k) \notin \Omega_{\text{syn}}$ necessarily comes from $\exists \models (\mathbb{K}, c_k)$. We can therefore conclude that $\exists \models \mathbb{K}\boxed{c_k}$, i.e. $\exists \models c$. ◀

Note that the proof for $c \not\rightharpoonup_{\Omega_{\text{sem}}}$ is even easier: We do not have $(\Box, c) \in \Omega_{\text{sem}}$ because otherwise we would have $c \rightharpoonup_{\Omega_{\text{sem}}} c$. We can therefore conclude that $\exists \models (\Box, c)$, and hence that $\exists \models c$.

The intuition behind the existence of a decidable bad set $\Omega$ is simple: In $\mathcal{L}_{\mathbf{c}}$ the equations imposed on the substitution by $\mathbb{K}$ are only of the shape $x^+ \sim \mathfrak{v}_k^\tau(\vec{a})$ and $\alpha^- \sim \mathfrak{s}_k^\tau(\vec{a})$ so that this is a first order unification problem and we can simply compute the most general unifier and apply it to the command. In the intuitionistic calculi, one can get equations that speak of $x^- V$ and things therefore get more complicated. In the $\mathcal{L}_{\mathbf{ivs}}$, there is always a single branch and no clashes are possible, so that $\Omega = \emptyset$ suffices. In $\mathcal{L}_{\mathbf{i}\not\Downarrow}$, one can get equations of the shape $x^- V_1 \sim \mathfrak{v}_1$ and $x^- V_2 \sim \mathfrak{v}_2$ so that one has to decide whether $V_1$ and $V_2$ are separable, and if so, substitute $x^-$ by the value that separates them. Fortunately, since both $V_i$ contain no negative value, and hence no command, deciding whether they are separable or not is easy. While we could therefore build one specific $\Omega$ per fragment, we prefer giving a unique $\Omega_{\text{syn}}$ for all those fragments, and a generic proof that it is decidable. The idea is that in all those fragments, one can bound the size of the substitution and the number of reduction steps needed as a function of $c$, so that $\exists \models c$, i.e. $\exists \varphi, c \rhd^* \langle x^\varepsilon \mid\mid \alpha^\varepsilon \rangle^\varepsilon$, becomes decidable. The extension to the decidability of $\exists \models (\mathbb{K}, c)$ is done by defining $\widehat{\mathbb{K}}$ as the same context where all branches not above the hole are replaced by $\Omega$ (or a clash, or some other unsolvable command whose shape is easy to detect). $\exists \models (\mathbb{K}, c)$ is then equivalent to $\exists \models \widehat{\mathbb{K}}\boxed{c}$.

▶ **Proposition 5.4.** *For any fixed $c$, $c'$ and $n$, the property "$\exists \varphi, c\,[\varphi] \rhd^n c'$" is decidable.*

**Proof.** In the appendix. ◀

## 6 Conclusion

▶ **Theorem 6.1.** *$\rightharpoonup_{\Omega_{syn}}$ operationally characterizes solvability in $\mathcal{L}_{\mathbf{c}}$ and $\mathcal{L}_{\mathbf{ivs}}$ and $\mathcal{L}_{\mathbf{i}\not\Downarrow}$.*

It seems plausible that $\rightharpoonup$ can be extended to the full intuitionistic calculus by making it reduce $V_1, \dots, V_n$ in parallel, whenever we detect that if none of these values are separable from some other values found in the command, then the term is not solvable. These reduction steps can a priori be postponed until after normal $\rightharpoonup$ steps without breaking substitutivity, so UTB should not be too hard. However, somewhere between solvability of normal forms and decidability of $\Omega_{\text{syn}}$, one would have to prove that for "normal-enough terms", separability is decidable, which we expect to be hard.

## 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*, pages 6–21, 2012.

2 Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. Distilling abstract machines (long version). *CoRR*, abs/1406.2370, 2014.

3 Beniamino Accattoli and Giulio Guerrieri. Open call-by-value. In *Programming Languages and Systems - 14th Asian Symposium, APLAS 2016, Hanoi, Vietnam, November 21-23, 2016, Proceedings*, pages 206–226, 2016.

4 Beniamino Accattoli and Giulio Guerrieri. Open call-by-value (extended version). *CoRR*, abs/1609.00322, 2016.

5 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*, pages 4–16, 2012.

6 H.P. Barendregt. *The lambda calculus: its syntax and semantics*. Studies in logic and the foundations of mathematics. North-Holland, 1984.

7 Harrie Jan Sander Bruggink. *Equivalence of Reductions in Higher-Order Rewriting*. PhD thesis, 2008.

8 Pierre-Louis Curien, Marcelo P. Fiore, and Guillaume Munch-Maccagnoni. A theory of effects and resources: adjunction models and polarised calculi. In Rastislav Bodík and Rupak Majumdar, editors, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pages 44–56. ACM, 2016.

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

10 Pierre-Louis Curien and Guillaume Munch-Maccagnoni. The duality of computation under focus. *CoRR*, abs/1006.2283, 2010.

11 Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. A new deconstructive logic: Linear logic. *J. Symb. Log.*, 62(3):755–807, 1997.

12 Paul Downen and Zena M. Ariola. A tutorial on computational classical logic and the sequent calculus. *J. Funct. Program.*, 28:e3, 2018.

13 Murdoch James Gabbay and Stéphane Lengrand. The lambda-context calculus (extended version). *Inf. Comput.*, 207(12):1369–1400, 2009.

14 Álvaro García-Pérez and Pablo Nogueira. No solvable lambda-value term left behind. *Logical Methods in Computer Science*, 12(2), 2016.

15 Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types*. Cambridge University Press, New York, NY, USA, 1989.

16 Hugo Herbelin. C'est maintenant qu'on calcule, 2005.

17 Jean-Louis Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 20(3):199–207, 2007.

18 Olivier Laurent. *A study of polarization in logic*. Theses, Université de la Méditerranée - Aix-Marseille II, March 2002.

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

20 Eugenio Moggi. Computational lambda-calculus and monads. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989*, pages 14–23. IEEE Computer Society, 1989.

21 Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, 1991.

22 Guillaume Munch-Maccagnoni. *Syntax and Models of a non-Associative Composition of Programs and Proofs. (Syntaxe et modèles d'une composition non-associative des programmes et des preuves)*. PhD thesis, Paris Diderot University, France, 2013.

23 Guillaume Munch-Maccagnoni. Note on Curry's style for Linear Call-by-Push-Value. working paper or preprint, May 2017.

24 Guillaume Munch-Maccagnoni. Note on models of polarised intuitionistic logic. working paper or preprint, June 2017.

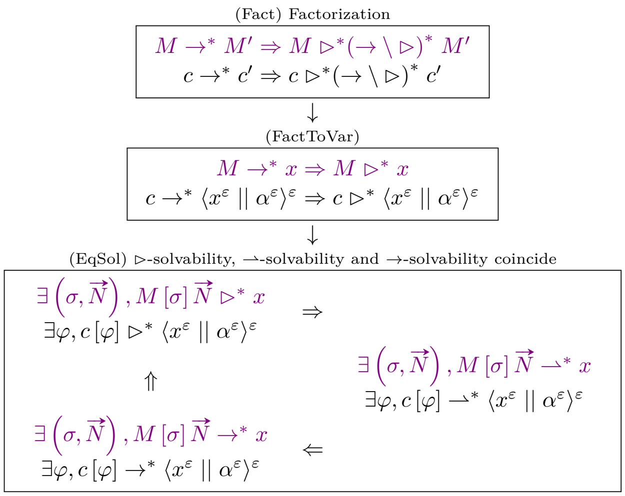


**Figure 5** Equivalence of solvability definitions - $\lambda$-calculus


**25** 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*, pages 127–140, 2015.

**26** Luca Paolini and Simona Ronchi Della Rocca. Call-by-value solvability. *ITA*, 33(6):507–534, 1999.

**27** Michel Parigot. Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction. In Andrei Voronkov, editor, *Logic Programming and Automated Reasoning,International Conference LPAR'92, St. Petersburg, Russia, July 15-20, 1992, Proceedings*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer, 1992.

**28** Christopher P. Wadsworth. The relation between computational and denotational properties for scott's $\mathrm{d}_{\mathrm{infty}}$-models of the lambda-calculus. *SIAM J. Comput.*, 5(3):488–521, 1976.


## Proofs of Section 2

▶ **Lemma .2.** *For any reductions $\rhd$ and $\rightharpoonup$ such that $\rhd \subseteq \rightharpoonup \subseteq \to$, if (Fact) and (RedToVar) then (EqSol). See Figure 5 on page 17.*

**Proof.** ◀

- $\boxed{(FactS)}$ Suppose that $c \to^* \langle x^\varepsilon \,||\, \alpha^\varepsilon \rangle^\varepsilon$. By (Fact), $c \rhd^* c' \,(\to \setminus \rhd)^n\, \langle x^\varepsilon \,||\, \alpha^\varepsilon \rangle^\varepsilon$ for some $n \in \mathbb{N}$. By (RedS), there is no $c''$ such that $c'' \,(\to \setminus \rhd)\, \langle x^\varepsilon \,||\, \alpha^\varepsilon \rangle^\varepsilon$, so that $n = 0$ and $c' = \langle x^\varepsilon \,||\, \alpha^\varepsilon \rangle^\varepsilon$. We can therefore conclude that $c \rhd^* \langle x^\varepsilon \,||\, \alpha^\varepsilon \rangle^\varepsilon$.
- $\boxed{(EqSol)}$ By $\rhd \subseteq \rightharpoonup \subseteq \to^*$, two of the implications are trivial, and the remaining one is (FactToVar).

Note that in order to show that $\rightharpoonup$-solvability is equivalent to solvability, it would be sufficient to show the following factorization: If $c \to^* c'$ then $c \rightharpoonup^* (\to \setminus \rightharpoonup)^* c'$. However, the $\rightharpoonup$ reduction will end up being far more complicated than the $\rhd$ one, so that the detour through $\rhd$-solvability as described in Figure 5 on page 17 actually simplifies proofs. An additional advantage of this proof is that we can pick $\rightharpoonup$ a posteriori, since we proved that $\rightharpoonup$-solvability is equivalent to solvability for any $\rightharpoonup$ such that $\rhd \subseteq \rightharpoonup \subseteq \to^*$.

In order to complete the proof, we need to prove (Fact) and (RedToVar). (RedToVar) is immediate. Note however that $(\lambda x.Ix) \,(\to \setminus \rhd)\, \lambda x.x$, which is another reason why we picked

$x$ and not $I$ in the definition of solvability. The only thing missing piece is the following factorization lemma (sometimes also called standardization, when presented in a slightly stronger form):

▶ **Lemma .3** (Factorization). *If $c \to^* c'$ then $c \rhd^* (\to \setminus \rhd)^* c'$.*

**Proof.** We apply a generic theorem for higher-order rewrite systems given in [7]. This theorem is stated below (with implicit hypothesis made explicit).

Another option would be to use [1]. ◀

▶ **Theorem .4** (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.*

## Proofs of Section 3

**of Lemma 3.7.** Suppose that $w \rightharpoonup w'$. By definition of $\rightharpoonup$, there exists a $l$-ahead multicontext $\mathbb{W}_0$ such that $w = \mathbb{W}_0\boxed{c_1, \dots, c_n}$, $w' = \mathbb{W}_0\boxed{c'_1, \dots, c'_n}$ and for each $k$, $c_k \overset{\mathbb{W}_0\boxed{\dots,c_{k-1},\Box,c_{k+1},\dots}}{\blacktriangleright}_{l,\Omega} c'_k$. By Lemma 3.2, $\mathbb{W}_0 [\varphi] = \mathbb{W}'_0\boxed{\Box^{\psi_1}, \dots, \Box^{\psi_n}}$ where $\mathbb{W}'_0$ is a $l$-ahead multicontext. For each $k$, since $(\mathbb{W}_0\boxed{\dots, c_{k-1}, \Box, c_{k+1}, \dots}) [\varphi] = \mathbb{W}'_0\boxed{\dots, c_{k-1} [\psi_{k-1}], \Box^{\psi_k}, c_{k+1} [\psi_{k+1}], \dots}$, by Lemma 3.6, $c_k [\psi_k] \overset{\mathbb{W}'_0\boxed{\dots,c_{k-1}[\psi_{k-1}],\Box,c_{k+1}[\psi_{k+1}],\dots}}{\blacktriangleright}_{l,\Omega} c'_k [\psi_k]$. We can therefore conclude that $w [\varphi] = \mathbb{W}'_0\boxed{c_1 [\psi_1], \dots, c_n [\psi_n]} \rightharpoonup_{l,\Omega} \mathbb{W}'_0\boxed{c_1 [\psi_1], \dots, c'_n [\psi_n]} = w' [\varphi]$. ◀

## Proofs of Section 4

**of Lemma 4.2.** By case analysis on the reduction $c_l \overset{\mathbb{K}}{\blacktriangleleft} c$.

- $\boxed{c_l = c \text{ and } (\mathbb{K}, c) \in \Omega}$ By (Bad-red-c), $(\mathbb{K}, c_r) \in \Omega$ so that we can take $c' = c_r$ and conclude that $c_l \overset{\mathbb{K}}{\underline{\rhd}} c' \overset{\mathbb{K}}{\blacktriangleleft} c_r$.
- $\boxed{c_l = c_k \left[\vec{A}/\vec{a_k}\right] \lhd \left\langle \mu\langle \mathfrak{s}_1^\tau (\vec{a_1}) .c_1 \mid \dots \mid \mathfrak{s}_n^\tau (\vec{a_n}) .c_n\rangle \mid\mid \mathfrak{s}_k^\tau \left(\vec{A}\right) \right\rangle^- = c}$ There are two possibilities for $c_r$:
  - $\boxed{c_r = \left\langle \mu\langle \mathfrak{s}_1^\tau (\vec{a_1}) .c'_1 \mid \dots \mid \mathfrak{s}_n^\tau (\vec{a_n}) .c'_n\rangle \mid\mid \mathfrak{s}_k^\tau \left(\vec{A}\right) \right\rangle^-}$ where for each $k$, $c_k \overset{\mathbb{K}\boxed{\mu\langle\dots|\mathfrak{s}_k^\tau(\vec{a_k}).\Box|\dots\rangle}}{\underline{\rhd}} c'_k$. We can pick $c' = c'_k \left[\vec{A}/\vec{a_k}\right]$. By Lemma 3.7, we have $c_l = c_k \left[\vec{A}/\vec{a_k}\right] \overset{\mathbb{K}}{\underline{\rhd}} c'_k \left[\vec{A}/\vec{a_k}\right] = c'$. We can therefore conclude that $c_l \overset{\mathbb{K}}{\underline{\rhd}} c' \lhd \left\langle \mu\langle \mathfrak{s}_1^\tau (\vec{a_1}) .c'_1 \mid \dots \mid \mathfrak{s}_n^\tau (\vec{a_n}) .c'_n\rangle \mid\mid \mathfrak{s}_k^\tau \left(\vec{A}\right) \right\rangle^-$.
  - $\boxed{c_r = \left\langle \mu\langle \mathfrak{s}_1^\tau (\vec{a_1}) .c_1 \mid \dots \mid \mathfrak{s}_n^\tau (\vec{a_n}) .c_n\rangle \mid\mid \mathfrak{s}_k^\tau \left(\vec{A'}\right) \right\rangle^-}$ This case is only possible in the intuitionistic version so that $\mathfrak{s}_k^\tau \left(\vec{A}\right) = \mathfrak{s}_k^\tau \left(\vec{V}, S\right)$, $\mathfrak{s}_k^\tau \left(\vec{A'}\right) = \mathfrak{s}_k^\tau \left(\vec{V}, S'\right)$ and $\vec{a_k} = \vec{x}, \star^\varepsilon$ with $S \overset{\langle\mu\dots||\mathfrak{s}_k^\tau(\vec{V},\Box)\rangle^-}{\underline{\rhd}} S'$. Let $c' = c_k \left[\vec{V}/\vec{x}, S'/\star^\varepsilon\right]$. By Lemma 4.1, we can therefore conclude that $c_l \overset{\mathbb{K}}{\underline{\rhd}} c' \lhd c_r$.
- The remaining cases are all similar to, and simpler than the previous case so we will not detail them.

◀

**of Lemma 4.3.** We show by induction on the syntax of $w$ that for any $w_l$, $w_r$, $w'_r$ and $\mathbb{K}$, if $w_l \xrightarrow{\mathbb{K}} w \xrightarrow{\mathbb{K}} w_r \xrightarrow{\mathbb{K}} w'_r$ then there exists $w'_l$ and $w'$ such that $w_l \xrightarrow{\mathbb{K}} w'_l \xrightarrow{\mathbb{K}} w' \xrightarrow{\mathbb{K}} w'_r$.

In some cases, we will instead prove a slightly stronger statement: There exists $w''$ such that $w_l \xrightarrow{\mathbb{K}} w'' \xrightarrow{\mathbb{K}} w_r$. We can get back the weaker result by taking $w'_l = w''$ and $w' = w_r$: $w_l \xrightarrow{\mathbb{K}} w'' \xrightarrow{\mathbb{K}} w_r \xrightarrow{\mathbb{K}} w'_r$.

- $\boxed{w \neq c}$ All cases where $w$ is not a command are done by applying the induction hypothesis, and the two most complex ones are $w = \tilde{\mu}\left[\mathfrak{v}_1^\tau\left(\overrightarrow{a_1}\right).c_1 \mid \ldots \mid \mathfrak{v}_n^\tau\left(\overrightarrow{a_n}\right).c_n\right]$ and $w = \mu\langle \mathfrak{s}_1^\tau\left(\overrightarrow{a_1}\right).c_1 \mid \ldots \mid \mathfrak{s}_n^\tau\left(\overrightarrow{a_n}\right).c_n\rangle$. Since they are similar, we only detail the $w = \tilde{\mu}\left[\mathfrak{v}_1^\tau\left(\overrightarrow{a_1}\right).c_1 \mid \ldots \mid \mathfrak{v}_n^\tau\left(\overrightarrow{a_n}\right).c_n\right]$ case. We have $w_l = \tilde{\mu}\left[\mathfrak{v}_1^\tau\left(\overrightarrow{a_1}\right).c_{1,l} \mid \ldots \mid \mathfrak{v}_n^\tau\left(\overrightarrow{a_n}\right).c_{n,l}\right]$, $w_r = \tilde{\mu}\left[\mathfrak{v}_1^\tau\left(\overrightarrow{a_1}\right).c_{1,r} \mid \ldots \mid \mathfrak{v}_n^\tau\left(\overrightarrow{a_n}\right).c_{n,r}\right]$ and $w'_r = \tilde{\mu}\left[\mathfrak{v}_1^\tau\left(\overrightarrow{a_1}\right).c'_{1,r} \mid \ldots \mid \mathfrak{v}_n^\tau\left(\overrightarrow{a_n}\right).c'_{n,r}\right]$ with for each $k$, $c_{k,l} \xrightarrow{\mathbb{K}\boxed{\tilde{\mu}[\ldots|\mathfrak{v}_k^\tau(\overrightarrow{a_k}).\square|\ldots]}} c_k \xrightarrow{\mathbb{K}\boxed{\tilde{\mu}[\ldots|\mathfrak{v}_k^\tau(\overrightarrow{a_k}).\square|\ldots]}} c_{k,r} \xrightarrow{\mathbb{K}\boxed{\tilde{\mu}[\ldots|\mathfrak{v}_k^\tau(\overrightarrow{a_k}).\square|\ldots]}} c'_{k,r}$. For each $k$, by the induction hypothesis, there exists $c'_{k,l}$ and $c'_k$ such that $c_{k,l} \xrightarrow{\mathbb{K}\boxed{\tilde{\mu}[\ldots|\mathfrak{v}_k^\tau(\overrightarrow{a_k}).\square|\ldots]}} c'_{k,l} \xrightarrow{\mathbb{K}\boxed{\tilde{\mu}[\ldots|\mathfrak{v}_k^\tau(\overrightarrow{a_k}).\square|\ldots]}} c'_k \xrightarrow{\mathbb{K}\boxed{\tilde{\mu}[\ldots|\mathfrak{v}_k^\tau(\overrightarrow{a_k}).\square|\ldots]}} c'_{k,r}$. We can therefore pick $w'_l = \tilde{\mu}\left[\mathfrak{v}_1^\tau\left(\overrightarrow{a_1}\right).c'_{1,l} \mid \ldots \mid \mathfrak{v}_n^\tau\left(\overrightarrow{a_n}\right).c'_{n,l}\right]$ and $w' = \tilde{\mu}\left[\mathfrak{v}_1^\tau\left(\overrightarrow{a_1}\right).c'_1 \mid \ldots \mid \mathfrak{v}_n^\tau\left(\overrightarrow{a_n}\right).c'_n\right]$, and conclude that $w_l \xrightarrow{\mathbb{K}} w'_l \xrightarrow{\mathbb{K}} w' \xrightarrow{\mathbb{K}} w'_r$.
- $\boxed{w = c}$ If $w$ is a command $c$, there there are several subcases depending on what the reductions are:
  - $\boxed{c_l = c \xrightarrow{\mathbb{K}} c_r \xrightarrow{\mathbb{K}} c'_r}$ We have $c_l = c$ and $(\mathbb{K}, c) \in \Omega$. By (Bad-red-c), $(\mathbb{K}, c_r) \in \Omega$ so that we can take $c'' = c_r$ and conclude that $c_l \xrightarrow{\mathbb{K}} c'' \overset{\mathbb{K}}{\blacktriangleleft} c_r$.
  - $\boxed{c_l \xrightarrow{\mathbb{K}} c = c_r \xrightarrow{\mathbb{K}} c'_r}$ We have $c = c_r$ and $(\mathbb{K}, c) \in \Omega$. By (Bad-red-c), $(\mathbb{K}, c_l) \in \Omega$ so that we can take $c'_l = c_l$ and $c' = c$ and get $c_l \overset{\mathbb{K}}{\blacktriangleright} c'_l \xrightarrow{\mathbb{K}} c = c_r \xrightarrow{\mathbb{K}} c'_r$.
  - $\boxed{c_l \lhd c \rhd c_r \xrightarrow{\mathbb{K}} c'_r}$ By 1.1, $c_l = c_r$ and we can therefore take $c'' = c'_r$ and conclude that $c_l \xrightarrow{\mathbb{K}} c'' \xrightarrow{\mathbb{K}} c_r$.
  - $\boxed{c_l \overset{\mathbb{K}}{\blacktriangleleft} c \left(\xrightarrow{\mathbb{K}} \setminus \overset{\mathbb{K}}{\blacktriangleright}\right) c_r \xrightarrow{\mathbb{K}} c'_r}$ By Lemma 4.2, there exists $c''$ such that $c_l \xrightarrow{\mathbb{K}} c'' \overset{\mathbb{K}}{\blacktriangleleft} c_r$ and we are done.
  - $\boxed{c_l \left(\xrightarrow{\mathbb{K}} \setminus \overset{\mathbb{K}}{\blacktriangleleft}\right) c \overset{\mathbb{K}}{\blacktriangleright} c_r \xrightarrow{\mathbb{K}} c'_r}$ By Lemma 4.2, there exists $c''$ such that $c_l \overset{\mathbb{K}}{\blacktriangleright} c'' \xrightarrow{\mathbb{K}} c_r$ and we are done.
  - $\boxed{c_l \left(\xrightarrow{\mathbb{K}} \setminus \overset{\mathbb{K}}{\blacktriangleleft}\right) c \left(\xrightarrow{\mathbb{K}} \setminus \overset{\mathbb{K}}{\blacktriangleright}\right) c_r \xrightarrow{\mathbb{K}} c'_r}$ There are two types of subcases: Either both reductions happen on the same side of the command, or they happen on different sides. We detail one of each.
    - $\boxed{\langle t \mid\mid e_l\rangle \xrightarrow{\mathbb{K}} \langle t \mid\mid e\rangle \xrightarrow{\mathbb{K}} \langle t_r \mid\mid e\rangle \xrightarrow{\mathbb{K}} c'_r}$ We can pick $c' = \langle t_r \mid\mid e_l\rangle^+$ and we are done because $\langle t \mid\mid e_l\rangle^+ \xrightarrow{\mathbb{K}} \langle t_r \mid\mid e_l\rangle^+ \xrightarrow{\mathbb{K}} \langle t_r \mid\mid e\rangle^+$. (Note that this case can only happen in the intuitionistic calculi.)
    - $\boxed{\langle V_l \mid\mid S\rangle^- \xrightarrow{\mathbb{K}} \langle V \mid\mid S\rangle^- \xrightarrow{\mathbb{K}} \langle V_r \mid\mid S\rangle^- \xrightarrow{\mathbb{K}} c'_r}$
      - $\boxed{c'_r = \langle V'_r \mid\mid S\rangle^-}$ We have $V_l \xrightarrow{\mathbb{K}_0} V \xrightarrow{\mathbb{K}_0} V_r \xrightarrow{\mathbb{K}_0} V'_r$, where $\mathbb{K}_0 = \mathbb{K}\boxed{\langle \square \mid\mid S\rangle^-}$. By the induction hypothesis, there exists $V'_l$ and $V'$ such that $V_l \xrightarrow{\mathbb{K}_0} V'_l \xrightarrow{\mathbb{K}_0} V' \xrightarrow{\mathbb{K}_0} V'_r$.

We can therefore pick $c'_l = \langle V'_l \mid\mid S\rangle^-$ and $c' = \langle V' \mid\mid S\rangle^-$ and we are done.

- $\boxed{c'_r = \langle V_r \mid\mid S'\rangle^-}$ We can take $c'_l = \langle V_l \mid\mid S'\rangle^-$ and $c' = \langle V \mid\mid S'\rangle^-$ and we are done.
- $\boxed{\langle V_r \mid\mid S\rangle^- \triangleright c'_r}$ We will detail the case $V = \mu\langle \mathfrak{s}^\tau_1(\overrightarrow{a_1}).c_1 \mid \ldots \mid \mathfrak{s}^\tau_n(\overrightarrow{a_n}).c_n\rangle$. The other cases are similar. We have $V_l = \mu\langle \mathfrak{s}^\tau_1(\overrightarrow{a_1}).c_{1,l} \mid \ldots \mid \mathfrak{s}^\tau_n(\overrightarrow{a_n}).c_{n,l}\rangle$, $V_r = \mu\langle \mathfrak{s}^\tau_1(\overrightarrow{a_1}).c_{1,r} \mid \ldots \mid \mathfrak{s}^\tau_n(\overrightarrow{a_n}).c_{n,r}\rangle$, $S_r = \mathfrak{s}^\tau_k\left(\overrightarrow{A}\right)$ and $c'_r = c_{k,r}\left[\overrightarrow{A}\middle/\overrightarrow{a_k}\right]$. We can therefore pick $c'_l = c_{k,l}\left[\overrightarrow{A}\middle/\overrightarrow{a_k}\right]$ and $c' = c_k\left[\overrightarrow{A}\middle/\overrightarrow{a_k}\right]$, and by Lemma 3.7, we have $c'_l \overset{\mathbb{K}}{\leftharpoondown} c'$ and we are done.

◀

▶ **Lemma .5.** *If $\sim$ is a bisimulation then so is $\sim^n$ for any $n$: If $c_l \sim^n c_r \rightharpoonup c'_r$ then there exists $c'_l$ such that $c_l \rightharpoonup c'_l \sim^n c'_r$.*

**Proof.** Since $\sim$ is a bisimulation, whenever $c \sim\rightharpoonup c'$, we have $c \rightharpoonup\sim c'$, i.e. we can postpone $\sim$ with respect to $\rightharpoonup$. We get the result by applying this $n$ times. ◀

▶ **Lemma .6.** *If $c_l \leftharpoonup^n c \rightharpoonup^n c_r$ then $c_l\ (\leftharpoonup\cdot\rightharpoonup)^n\ c_r$.*

**Proof.** By induction on $n$. The base case is trivial. In the inductive case, we have $c'_l \leftharpoonup c_l \leftharpoonup^n c \rightharpoonup^n c_r \rightharpoonup c'_r$. By the induction hypothesis, $c_l\ (\leftharpoonup\cdot\rightharpoonup)^n\ c_r$. By lemma .5 and $c'_l \leftharpoonup c_l\ (\leftharpoonup\cdot\rightharpoonup)^n\ c_r$, we get that there exists $c''_r$ such that $c'_l\ (\leftharpoonup\cdot\rightharpoonup)^n\ c''_r \leftharpoonup c_r$. We therefore have $c'_l\ (\leftharpoonup\cdot\rightharpoonup)^n\ c''_r \leftharpoonup c_r \rightharpoonup c'_r$, i.e. $c'_l\ (\leftharpoonup\cdot\rightharpoonup)^{n+1}\ c_r$. ◀

▶ **Lemma .7.** *If $\not\bowtie c_l \leftharpoonup^n c \rightharpoonup^n c_r$ then $c_r \not\bowtie$.*

**Proof.** By lemma .6, we have $\not\bowtie c_l\ (\leftharpoonup\cdot\rightharpoonup)^n\ c_r$. If we had $c_r \rightharpoonup$ then by lemma .5, we would have $c_l \rightharpoonup$ which is absurd. ◀

**of lemma 4.4.** Suppose that $\not\bowtie c_l \leftharpoonup^n c \rightharpoonup^\omega$. There exists $c_r$ such that the reduction $c \rightharpoonup^\omega$ is of the shape $c \rightharpoonup^n c_r \rightharpoonup^\omega$. Since $\not\bowtie c_l \leftharpoonup^n c \rightharpoonup^n c_r$, by lemma .7 $c_r \not\bowtie$ which is absurd. ◀

## Proofs of Section 5

▶ **Definition .8.** $\sim_d :=$ equal up to depth $d$

($\sim_d$-contr)

$$\boxed{d_1 \leq d_2 \Rightarrow \sim_{d_1} \supseteq \sim_{d_2}}$$

($\sim_d$-subst)

$$\boxed{c \sim_d c' \Rightarrow c[\varphi] \sim_d c'[\varphi]}$$

($\sim_d$-in-subst)

$$\boxed{\varphi \sim_d \varphi' \Rightarrow c[\varphi] \sim_d c[\varphi']}$$

($\sim_{d+n}\triangleright^n$-swap)

$$\boxed{c \sim_{d+n}\triangleright^n c' \Rightarrow c \triangleright^n \sim_d c'}$$

($c\sim_{\updownarrow c \updownarrow}$-singleton)

$$\boxed{c \sim_{\updownarrow c \updownarrow} c' \Rightarrow c = c'}$$

($\triangleright^d$-truncate)

$$\boxed{c[\varphi] \triangleright^n c' \Rightarrow \exists \varphi', \updownarrow\varphi'\updownarrow \leq \updownarrow c'\updownarrow + n \wedge c[\varphi'] \triangleright^n c'}$$

(Subst-truncate)

$$\boxed{\forall d, \forall \varphi, \exists \varphi', \updownarrow\varphi'\updownarrow \leq d \wedge \varphi' \sim_d \varphi}$$

▶ **Lemma .9.** *If $c[\varphi] \triangleright^d c'$ then there exists $\varphi'$ such that $\updownarrow\varphi'\updownarrow \leq d$ and $c[\varphi'] \triangleright^d c'$.*

**Proof.** See figure.

- $\boxed{\sim_d \text{-contr}}$ By induction on $d_1$.
- $\boxed{\sim_d \text{-subst}}$ By induction on $d$, using ($\sim_d$ -contr).
- $\boxed{\sim_d \text{-in-subst}}$ By induction on $d$, using ($\sim_d$ -contr).
- $\boxed{\sim_{d+n}\rhd^n \text{-swap}}$ It is sufficient to show this for $n = 1$, as the general case can then be proved by induction on $n$. Suppose that $c \sim_{d+1} c' \rhd c''$. We show that $c \rhd\sim_d c''$ by case analysis on the reduction $c' \rhd c''$.
  - $\boxed{c' = \langle \mu\alpha^\varepsilon.c'_0 \mid\mid S'_\varepsilon\rangle^\varepsilon \rhd c'_0 \left[S'_\varepsilon/\alpha^\varepsilon\right] = c''}$ Since $c \sim_{d+1} c'$, there exists $c_0$ and $S_\varepsilon$ such that $c_0 \sim_d c'_0$, $S_\varepsilon \sim_{d+1} S'_\varepsilon$ and $c = \langle \mu\alpha^\varepsilon.c_0 \mid\mid S_\varepsilon\rangle^\varepsilon$. We can therefore conclude $c = \langle \mu\alpha^\varepsilon.c_0 \mid\mid S_\varepsilon\rangle^\varepsilon \rhd c_0 \left[S_\varepsilon/\alpha^\varepsilon\right] \sim_d c'_0 \left[S'_\varepsilon/\alpha^\varepsilon\right] = c''$ by ($\sim_d$ -subst) and ($\sim_d$ -in-subst).
  - The remaining cases are similar.
- $\boxed{c \sim_{\updownarrow c \updownarrow} \text{-singleton}}$ By induction on the syntax of $c$.
- $\boxed{\text{Subst-truncate}}$ It is sufficient to prove that: (Expr-truncate) For any $d$ and $w$, there exists $w'$ such that $\updownarrow w' \updownarrow \leq d$ and $w' \sim_d w$. (Subst-truncate) is then obtained by taking for each $a$, $\varphi'(a) := w'$ where $w'$ is the result of (Expr-truncate) for $w = \varphi(a)$. (Expr-truncate) is shown by induction on $d$.
  - $\boxed{d = 0}$ Let $w'$ be the result of replacing all $t_\varepsilon$ by $\alpha^\varepsilon$ and $e_\varepsilon$ by $x^\varepsilon$ in $w$. We have $\updownarrow w' \updownarrow = 0 \leq d$ and $w' \sim_0 w$ because $\sim_0$ identifies all terms.
  - $\boxed{d = d_0 + 1}$ Sufficient case analysis (to be able to get subexpressions that are $\sim_{d_0}$), and then applying the induction hypothesis works.
- $\boxed{\rhd^d \text{-truncate}}$ Suppose that $c[\varphi] \rhd^n c'$. By (Subst-truncate), there exists $\varphi'$ such that $\updownarrow \varphi' \updownarrow \leq \updownarrow c' \updownarrow + n$ and $\varphi' \sim_{\updownarrow c' \updownarrow + n} \varphi$. By ($\sim_d$ -in-subst), we have $c[\varphi'] \sim_{\updownarrow c' \updownarrow + n} c[\varphi]$. We therefore have $c[\varphi'] \sim_{\updownarrow c' \updownarrow + n} c[\varphi] \rhd^n c'$, and hence, by ($\sim_{d+n}\rhd^n$ -swap), there exists $c''$ such that $c[\varphi'] \rhd^n c'' \sim_{\updownarrow c' \updownarrow} c'$. By $\left(c \sim_{\updownarrow c \updownarrow} \text{-singleton}\right)$, $c'' = c'$. We can therefore conclude that $c[\varphi'] \rhd^n c'$ where $\updownarrow \varphi' \updownarrow \leq \updownarrow c' \updownarrow + n$.

◀

**of 5.4.** By Lemma .9, this property is equivalent to "$\exists \varphi, \updownarrow \varphi \updownarrow \leq \updownarrow c' \updownarrow + n \wedge c[\varphi] \rhd^n c'$", which is decidable because there are only finitely many substitutions $\varphi$ of height $\updownarrow \varphi \updownarrow$ bounded by $\updownarrow c' \updownarrow + n$, and finitely many $\rhd$ reduction paths of length bounded by $n$. ◀

## Proof of Lemma 2.6

**Proof.** The structure is described in Figure 6 on page 22.

- $\boxed{\text{(WNSol)}}$ If $c \rightharpoonup^* c' \not\rightharpoonup$ then by (NFSol), there exists $\varphi$ such that $c'[\varphi] \rightharpoonup^* \langle x^\varepsilon \mid\mid \alpha^\varepsilon\rangle^\varepsilon$. By (Subst), we have $c[\varphi] \rightharpoonup^* c'[\varphi]$ and we can therefore conclude that $c[\varphi] \rightharpoonup^* \langle x^\varepsilon \mid\mid \alpha^\varepsilon\rangle^\varepsilon$.
- $\boxed{\text{(SubstSN)}}$ The contrapositive is a corollary of (Subst).
- $\boxed{\text{(SolSN)}}$ If $M[\sigma]\vec{N} \rightharpoonup^* \mathcal{S}$ then by (UTB), we have $M[\sigma]\vec{N} \not\rightharpoonup^\infty$. By (SubstSN), we can therefore conclude that $M \not\rightharpoonup^\infty$.
- $\boxed{\text{(OpCharAhead)}}$ (WNSol) and (SolSN) give two of the implications, and the third one (that strongly-normalizing implies weakly-normalizing) is well-known.

◀

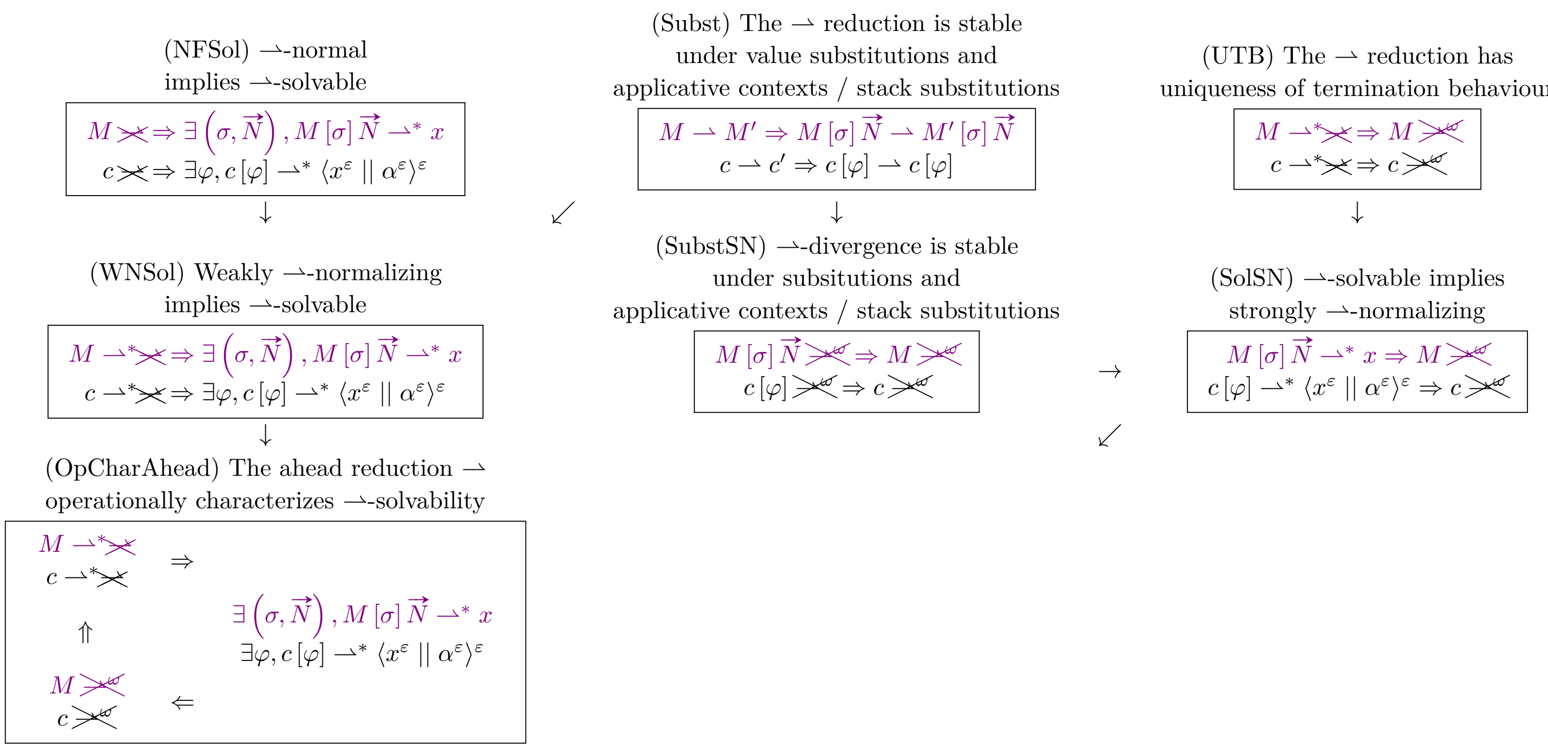


**Figure 6** Proof structure of Lemma 2.6