Logical Methods in Computer Science
Volume 20, Issue 2, 2024, pp. 4:1–4:52
https://lmcs.episciences.org/

Submitted Jan. 15, 2021
Published Apr. 18, 2024

A STRONG BISIMULATION FOR A CLASSICAL TERM CALCULUS

EDUARDO BONELLI [a], DELIA KESNER [b], AND ANDRÉS VISO [c]

[a] Stevens Institute of Technology, USA
e-mail address: eabonelli@gmail.com

[b] Université Paris Cité, CNRS, IRIF
e-mail address: kesner@irif.fr

[c] Inria, France
e-mail address: andres-ezequiel.viso@inria.fr

Abstract. When translating a term calculus into a graphical formalism many inessential details are abstracted away. In the case of λ-calculus translated to proof-nets, these inessential details are captured by a notion of equivalence on λ-terms known as $\simeq_\sigma$-equivalence, in both the intuitionistic (due to Regnier) and classical (due to Laurent) cases. The purpose of this paper is to uncover a strong bisimulation behind $\simeq_\sigma$-equivalence, as formulated by Laurent for Parigot's $\lambda\mu$-calculus. This is achieved by introducing a relation $\simeq$, defined over a revised presentation of $\lambda\mu$-calculus we dub ΛM.

More precisely, we first identify the reasons behind Laurent's $\simeq_\sigma$-equivalence on $\lambda\mu$-terms failing to be a strong bisimulation. Inspired by Laurent's *Polarized Proof-Nets*, this leads us to distinguish multiplicative and exponential reduction steps on terms. Second, we enrich the syntax of $\lambda\mu$ to allow us to track the exponential operations. These technical ingredients pave the way towards a strong bisimulation for the classical case. We introduce a calculus ΛM and a relation $\simeq$ that we show to be a strong bisimulation with respect to reduction in ΛM, *i.e.* two $\simeq$-equivalent terms have the exact same reduction semantics, a result which fails for Regnier's $\simeq_\sigma$-equivalence in λ-calculus as well as for Laurent's $\simeq_\sigma$-equivalence in $\lambda\mu$. Although $\simeq$ is formulated over an enriched syntax and hence is not strictly included in Laurent's $\simeq_\sigma$, we show how it can be seen as a restriction of it.

1. Introduction

An important topic in the study of programming language theories is unveiling structural similarities between expressions. They are widely known as *structural equivalences*; equivalent expressions behaving exactly in the same way. Process calculi are a rich source of examples. In CCS, expressions stand for processes in a concurrent system. For example, $P \parallel Q$ denotes the parallel composition of processes P and Q. Structural equivalence includes equations such as the one stating that $P \parallel Q$ and $Q \parallel P$ are equivalent. This minor reshuffling of subexpressions has little impact on the behavior of the overall expression: structural equivalence is a *strong bisimulation* for process reduction.

This paper is concerned with such notions of reshuffling of expressions in λ*-calculi with control operators*. The induced notion of structural equivalence, in the sequel $\simeq$, should

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.46298/LMCS-20(2:4)2024

identify terms having exactly the same reduction semantics too. Stated equivalently, $\simeq$ should be a strong bisimulation with respect to reduction in these calculi. This means that $\simeq$ should be symmetric and moreover $o \simeq p$ and $o \rightsquigarrow o'$ should imply the existence of p' such that $p \rightsquigarrow p'$ and $o' \simeq p'$, where $\rightsquigarrow$ denotes some given notion of reduction for control operators. Graphically,

$$\begin{array}{ccc} o & \simeq & p \\ \Big\downarrow\!\wr & & \Big\downarrow\!\wr \\ o' & \simeq & p' \end{array} \tag{1.1}$$

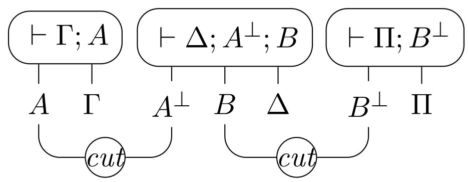
It is worth mentioning that we are not after a general theory of program equivalence. On the one hand, not all terms having the same reduction semantics are identified, only those resulting from reshuffling in the sense made precise below. On the other hand, there are terms that do not have the same reduction semantics but would still be considered to "behave in the same way" (*e.g.* (1.2) below). In particular, our proposed notion of equivalence is not a bisimilarity: there are terms that have the same reduction behavior but are not related by our $\simeq$-equivalence.

Before addressing λ-calculi with control operators, we comment on the state of affairs in the λ-calculus. Formulating structural equivalences for the λ-calculus is hindered by the sequential (left-to-right) orientation in which expressions are written. Consider for example the terms $(\lambda x.(\lambda y.t)\,u)\,v$ and $(\lambda x.\lambda y.t)\,v\,u$. They seem to have the same redexes, only permuted, similar to the situation captured by the above mentioned CCS equation. A closer look, however, reveals that this is not entirely correct. The former has two redexes (one indicated below by underlining and another by overlining) and the latter has only one (underlined):

$$\underline{(\lambda x.\overline{(\lambda y.t)\,u})\,v} \text{ and } \underline{(\lambda x.(\lambda y.t))\,v}\,u \tag{1.2}$$

The overlined redex on the left-hand side is not visible on the right-hand side; it will only reappear, as a newly *created* redex, once the underlined redex is computed. Despite the fact that the syntax gets in the way, Regnier [Reg94] proved that these terms behave in *essentially* the same way. More precisely, he introduced a structural equivalence for λ-terms, known as *σ-equivalence* and proved that σ-equivalent terms have head, leftmost, perpetual and, more generally, maximal reductions of the same length. However, the mismatch between the terms in (1.2) is unsatisfying since there clearly seems to be an underlying strong bisimulation, which is not showing itself due to a notational shortcoming. It turns out that through the graphical intuition provided by linear logic *proof-nets* (PN), one can define an enriched λ-calculus with explicit substitutions (ES) that unveils a strong bisimulation for the intuitionistic case [ABKL14]. In this paper, we resort to this same intuition to explore whether it is possible to uncover a strong bisimulation behind the notion of σ-equivalence formulated by Laurent [Lau02, Lau03] in the setting of classical logic. Thus, we will not only capture structural equivalence on pure functions, but also on *programs with control operators*. We next briefly revisit proof-nets and discuss how they help unveil structural equivalence as strong bisimulation for λ-calculi. An explanation of the challenges that we face in addressing the classical case will follow.

Proof-nets. A proof-net is a graph-like structure whose nodes denote logical inferences and whose edges or wires denote the formula they operate on. Proof-nets were introduced in the setting of linear logic [Gir87], a logic that provides a mechanism to explicitly control the use of resources by restricting the application of the *structural* rules of weakening and contraction. Proof-nets are equipped with an operational semantics specified by graph transformation rules which captures cut elimination in sequent calculus. The resulting cut elimination rules on proof-nets are split into two different kinds: *multiplicative*, that essentially reconfigure wires, and *exponential*, which are the only ones that are able to erase or duplicate (sub)proof-nets. Most notably, proof-nets abstract away the order in which certain rules occur in a sequent derivation. As an example, assume three derivations of the judgements $\vdash \Gamma; A$, $\vdash \Delta; A^\perp; B$ and $\vdash \Pi; B^\perp$, resp. The order in which these derivations are composed via cuts into a single derivation is abstracted away in the resulting proof-net:



In other words, *different* terms/derivations are represented by the *same* proof-net. Hidden structural similarity between terms can thus be studied by translating them to proof-nets. Moreover, following the Curry–Howard isomorphism which relates computation and logic, this correspondence between a term language and a graphical formalism can also be extended to their reduction behavior [Acc18, Kes22]. In this paper we focus on defining one such structural equality that is a strong bisimulation for a classical lambda calculus based on Parigot's $\lambda\mu$-calculus [Par92, Par93]. Although we rely on intuitions provided by Laurent's Polarized Proof Nets [Lau02, Lau03], knowledge about Polarized Proof Nets is not required to read this work and is not further discussed. We begin with an overview of a similar program carried out in the intuitionistic case.

Intuitionistic σ-Equivalence. Regnier introduced a notion of *σ-equivalence* on λ-terms (written $\simeq_\sigma$ and depicted in Figure 1), and proved that σ-equivalent terms behave in essentially identical way. This equivalence relation involves permuting certain redexes, and was unveiled through the study of proof-nets. In particular, following Girard's encoding of intuitionistic logic into linear logic [Gir87], σ-equivalent terms are mapped to the same proof-net (modulo multiplicative cuts and structural equivalence of PN).

$$
\begin{array}{rcll}
(\lambda x.\lambda y.t)\, u & \simeq_{\sigma_1} & \lambda y.(\lambda x.t)\, u & y \notin u \\
(\lambda x.t\, v)\, u & \simeq_{\sigma_2} & (\lambda x.t)\, u\, v & x \notin v
\end{array}
$$

Figure 1: σ-Equivalence for λ-terms

The reason why Regnier's result is not immediate is that redexes present on one side of an equation may disappear on the other side of it, as illustrated with the terms in (1.2). One might rephrase this observation by stating that $\simeq_\sigma$ is *not a strong bisimulation* over the set of λ-terms. If it were, then establishing that σ-equivalent terms behave essentially in the same way would be trivial.

Adopting a more refined view of λ-calculus as suggested by linear logic, which splits cut elimination on logical derivations into multiplicative and exponential steps, yields a decomposition of β-reduction on terms into multiplicative/exponential steps. The theory of *explicit substitutions* (a survey can be found in [Kes09]) provides a convenient syntax to reflect this splitting at the term level. Indeed, β-reduction can be decomposed into two steps, namely $\mathtt{B}$ (for $\mathtt{B}$eta), and $\mathtt{S}$ (for $\mathtt{S}$ubstitution):

$$\begin{array}{rcl} (\lambda x.t)\,u & \mapsto_{\mathtt{B}} & t[x\backslash u] \\ t[x\backslash u] & \mapsto_{\mathtt{S}} & t\{x\backslash u\} \end{array} \tag{1.3}$$

or, more generally, the *reduction at a* $\mathtt{d}$*istance* version for $\mathtt{B}$, introduced in [AK10], and written $\mathtt{dB}$:

$$\begin{array}{rcl} (\lambda x.t)[x_1\backslash v_1]\ldots[x_n\backslash v_n]\,u & \mapsto_{\mathtt{dB}} & t[x\backslash u][x_1\backslash v_1]\ldots[x_n\backslash v_n] \\ t[x\backslash u] & \mapsto_{\mathtt{S}} & t\{x\backslash u\} \end{array} \tag{1.4}$$

Firing the $\mathtt{dB}$-rule creates a new *explicit substitution* operator, written $[x\backslash u]$, so that $\mathtt{dB}$ essentially reconfigures symbols (it is in some sense an innocuous or *plain* rule), and indeed reads as a multiplicative cut in proof-nets. The $\mathtt{S}$-rule executes the substitution by performing a replacement of all free occurrences of x in t with u, written $t\{x\backslash u\}$, so that it is $\mathtt{S}$ that performs interesting or *meaningful* computation in the sense that it performs exponential cut steps in proof-nets. We write $\to_{\mathtt{S}}$ for $\mapsto_{\mathtt{S}}$ steps inside an arbitrary context and similarly for $\to_{\mathtt{dB}}$.

Decomposition of β-reduction by means of the reduction rules in (1.4) prompts one to replace Regnier's $\simeq_\sigma$ (Figure 1) with a new relation [AK12b] that we write here $\simeq_{\tilde\sigma}$ (Figure 2). The latter is formed essentially by taking the $\mathtt{dB}$-*normal form* of each side of the $\simeq_\sigma$ equations. Also included in $\simeq_{\tilde\sigma}$ is a third equation $\simeq_{\tilde\sigma_3}$ allowing commutation of orthogonal (independent) substitutions. Notice however that the $\mathtt{dB}$-expansion of $\simeq_{\tilde\sigma_3}$ results in σ-equivalent terms, since $t[y\backslash v][x\backslash u] \simeq_{\tilde\sigma} t[x\backslash u][y\backslash v]$, with $x \notin v$ and $y \notin u$, $\mathtt{dB}$-expands to $(\lambda y.(\lambda x.t)\,u)\,v \simeq_\sigma (\lambda x.(\lambda y.t)\,v)\,u$, both of which are σ-equivalent by $\simeq_{\sigma_1}$ and $\simeq_{\sigma_2}$.

$$\begin{array}{rcll} (\lambda y.t)[x\backslash u] & \simeq_{\tilde\sigma_1} & \lambda y.t[x\backslash u] & y \notin u \\ (t\,v)[x\backslash u] & \simeq_{\tilde\sigma_2} & t[x\backslash u]\,v & x \notin v \\ t[y\backslash v][x\backslash u] & \simeq_{\tilde\sigma_3} & t[x\backslash u][y\backslash v] & x \notin v, y \notin u \end{array}$$

Figure 2: Strong Bisimulation $\simeq_{\tilde\sigma}$ for λ-Terms

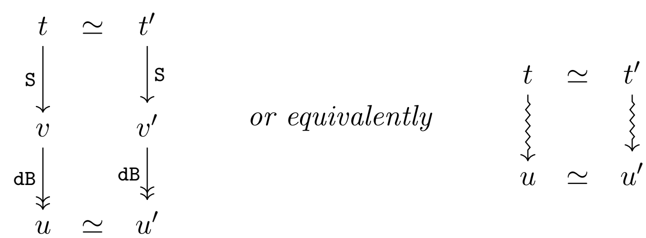
Through $\simeq_{\tilde\sigma}$ it is possible to unveil a strong bisimulation for the intuitionistic case by working on λ-terms with ES and the notion of β-reduction at a distance. Indeed, the following holds:

Theorem 1.1 (Strong Bisimulation for the Intuitionistic Case I)**.** *Let* $t \simeq_{\tilde\sigma} t'$*. If* $t \to_{\mathtt{dB},\mathtt{S}} u$*, then there exists* u' *such that* $t' \to_{\mathtt{dB},\mathtt{S}} u'$ *and* $t' \simeq_{\tilde\sigma} u'$*. Graphically,*

$$\begin{array}{ccc} t & \simeq_{\tilde\sigma} & t' \\ {\scriptstyle \mathtt{dB},\mathtt{S}}\downarrow & & \downarrow{\scriptstyle \mathtt{dB},\mathtt{S}} \\ u & \simeq_{\tilde\sigma} & u' \end{array}$$

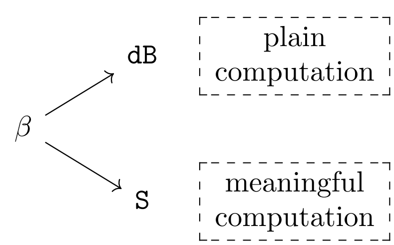
While any two $\simeq_{\tilde\sigma}$-equivalent λ-terms with ES translate to the same proof-net, the converse is not true. For example, the terms $(\lambda x.t\,v)\,u$ and $(\lambda x.t)\,u\,v$, where $x \notin v$, translate to the same proof-net [Reg94] (indeed, they are σ-equivalent), however they are not $\simeq_{\tilde\sigma}$-equivalent. Still, as remarked in [AK12b], the $\mathtt{dB}$-normal forms of those terms, namely $(t\,v)[x\backslash u]$ and $t[x\backslash u]\,v$, are $\simeq_{\tilde\sigma}$-equivalent, thus suggesting an alternative equivalence relation that we define only on dB-normal forms. Indeed, *plain forms* are λ-terms with ES that are in dB-normal form (which are intuitively, those terms that are in multiplicative normal form from the proof-net point of view). Similarly, *plain computation* is given by dB-reduction; plain computation to normal form produces plain forms. As mentioned above our notion of *meaningful computation* is taken to be the reduction relation $\rightarrow_{\mathtt{S}}$. We write $\simeq$ for this refined equivalence notion on plain forms, which will in fact be included in the strong bisimulation that we propose in this paper (*cf.* Definition 5.5). The following variation of the theorem mentioned above is obtained:

Theorem 1.2 (Strong Bisimulation for the Intuitionistic Case II)**.** *Let $\rightsquigarrow$ be $\rightarrow_{\mathtt{S}}$ followed by $\rightarrow_{\mathtt{dB}}$-reduction to* dB*-normal form. Let t, t' be two terms in* dB*-normal form such that $t \simeq t'$. If $t \rightsquigarrow u$, then there exists u' such that $t' \rightsquigarrow u'$ and $t' \simeq u'$. Graphically,*



Thus, a strong bisimulation can be defined on a set of *plain forms* (here λ-terms with ES which are in dB-normal form), with respect to a *meaningful computation* relation (here $\rightarrow_{\mathtt{S}}$) followed by dB-reduction to dB-normal form.

Summing up, a strong bisimulation was obtained by decomposing β-reduction as follows:



This methodology consisting in identifying an appropriate notion of plain form and meaningful computation, both over terms, allows for a strong bisimulation to surface. This requires establishing a corresponding distinction between multiplicative and exponential steps in the underlying term semantics. We propose following this same methodology for the classical case. However, as we will see, the notion of plain computation as well as that of meaningful one are not so easy to construct for Parigot's $\lambda\mu$-calculus. We next briefly introduce this calculus as well as the notion of σ-equivalence as presented by Laurent.

Classical σ-Equivalence. *λ-calculi with control operators* include operations to manipulate the context in which a program is executed. We focus here on Parigot's $\lambda\mu$-calculus, which extends the λ-calculus with two new operations: $[\alpha]\,t$ (*command*) and $\mu\alpha.c$ (*μ-abstraction*). The former may informally be understood as "call continuation α with t as argument" and the latter as "record the current continuation as α and continue as c". Reduction in $\lambda\mu$ consists of the β-rule together with:

$$(\mu\alpha.c)\,u \quad \mapsto_{\mu} \quad \mu\alpha.c\{\!\{\alpha \setminus u\}\!\}$$

where $c\{\!\{\alpha \setminus u\}\!\}$, called here *replacement*, replaces all subexpressions of the form $[\alpha]\,t$ in c with $[\alpha]\,(t\,u)$.

Regnier's notion of σ-equivalence for λ-terms was extended to $\lambda\mu$ by Laurent [Lau03] (*cf.* Figure 3 in Section 3). Here is an example of terms related by this extension, where the redexes are underlined/overlined and $\simeq_\sigma$ denotes Laurent's aforementioned relation:

$$(\underline{(\lambda x.\mu\alpha.[\gamma]\,u)\,w})\,v \simeq_\sigma \overline{(\mu\alpha.[\gamma]\,\underline{(\lambda x.u)\,w})\,v}$$

Once again, the fact that a harmless permutation of redexes has taken place is not obvious. The term on the right has two redexes (μ and β) but the one on the left only has one (β) redex. Another, more subtle, example of terms related by Laurent's extension clearly suggests that operational indistinguishability cannot rely on relating arbitrary μ-redexes; the underlined μ-redex on the left does not appear at all on the right:

$$\underline{(\mu\alpha.[\alpha]\,x)\,y} \simeq_\sigma x\,y \tag{1.5}$$

Clearly, Laurent's σ-equivalence on $\lambda\mu$-terms *fails to be a strong bisimulation.*

Towards a Strong Bisimulation for $\lambda\mu$. We seek to formulate a similar notion of equivalence for calculi with control operators in the sense that it is concerned with harmless permutation of redexes possibly involving control operators and induces a strong bisimulation. A first step towards our goal involves decomposing the μ-rule as was done for the β-rule in (1.4):

$$\begin{array}{rcl} (\mu\alpha.c)\,u & \mapsto & \mu\alpha'.c[\![\alpha \setminus^{\alpha'} u]\!] \\ c[\![\alpha \setminus^{\alpha'} u]\!] & \mapsto & c\{\!\{\alpha \setminus^{\alpha'} u\}\!\} \end{array} \tag{1.6}$$

where $c\{\!\{\alpha \setminus^{\alpha'} u\}\!\}$ denotes the *fresh* replacement changing all subexpressions of the form $[\alpha]\,t$ in c to $[\alpha']\,(tu)$. A brief discussion on our choice of notation for replacement, may be found at the end of this section. We still need to add the notion of distance to this operational semantics, as done for substitution. This produces a rule `dM` (for `M`*u* at a `d`*istance*), to introduce an *explicit replacement*, and another rule `R` (for `R`eplacement), that executes explicit replacements:

$$\begin{array}{rcl} (\mu\alpha.c)[x_1 \setminus v_1]\ldots[x_n \setminus v_n]\,u & \mapsto_{\mathtt{dM}} & (\mu\alpha'.c[\![\alpha \setminus^{\alpha'} u]\!])[x_1 \setminus v_1]\ldots[x_n \setminus v_n] \\ c[\![\alpha \setminus^{\alpha'} u]\!] & \mapsto_{\mathtt{R}} & c\{\!\{\alpha \setminus^{\alpha'} u\}\!\} \end{array} \tag{1.7}$$

where $c\{\!\{\alpha \setminus^{\alpha'} u\}\!\}$ replaces each sub-expression of the form $[\alpha]\,t$ in c by $[\alpha']\,(tu)$.

Following our analogy with the intuitionistic case, our plain rule is `dM` and meaningful computation is performed by `R`.

Therefore, we tentatively fix our notion of meaningful computation to be $\mathtt{S} \cup \mathtt{R}$ over the set of plain forms, the latter now obtained by taking *both* `dB` and `dM`-normal forms. However, in contrast to the intuitionistic case where the decomposition of β into a multiplicative rule `dB` and an exponential rule `S` suffices for unveiling the strong bisimulation behind Regnier's σ-equivalence in λ-calculus, it turns out that splitting the μ-rule into `dM` and `R` is not enough in the classical case. We face two obstacles:

Decomposing Meaningful Steps: Consider (1.5) from above. The methodology that led [AK12b] to obtain the theory of Figure 2 by taking the plain forms of Regnier's σ-equivalence, would lead us to the equation $\mu\alpha'.([\alpha]\,x)[\![\alpha \setminus^{\alpha'} y]\!] \simeq x\,y$, where plain forms are terms in $\mathtt{dB} \cup \mathtt{dM}$-normal form. However, there is clearly an `R` step on the left term which is not present on the right one:

$$\begin{array}{ccc} \mu\alpha'.([\alpha]\,x)[\![\alpha \setminus^{\alpha'} y]\!] & \simeq & x\,y \\ \Big\downarrow{\scriptstyle \mathtt{R}} & & \not\Big\downarrow{\scriptstyle \mathtt{R}} \\ \mu\alpha'.[\alpha']\,x\,y & \simeq & x\,y \end{array} \tag{1.8}$$

In its full generality, the R rule in (1.7) can certainly duplicate or erase u. However, it may also be the case that there is a unique occurrence of α in c. If, furthermore, this occurrence cannot be duplicated or erased any time later, then this instance of R may quite reasonably be catalogued as non-meaningful. Indeed, these *linear* replacements will form part of our notion of plain computation rather than that of the meaningful one. With this revised notion of plain computation, the plain normal form of $(\mu\alpha.[\alpha]\,x)\,y$ becomes $\mu\alpha'.[\alpha']\,x\,y$ and thus equation (1.5) now reads: $\mu\alpha'.[\alpha']\,x\,y \simeq x\,y$; moreover both terms are indeed related in the strong bisimulation relation that we propose in this paper.

Name Renaming: In $\lambda\mu$, an expression such as $[\alpha]\,\mu\beta.c$ may be simplified to another expression $c\{\beta \setminus \alpha\}$ where all occurrences of the name β in c are replaced with α. The effect is thus to rename β with α in c. Such an equation, dubbed $\simeq_\rho$, is included in Laurent's $\simeq_\sigma$ and also breaks strong bisimulation.

$$\begin{array}{ccc} (\mu\alpha.[\alpha]\,\mu\beta.[\gamma]\,x)\,u & \simeq_\rho & (\mu\alpha.[\gamma]\,x)\,u \\ \Big\downarrow{\scriptstyle \mu} & & \Big\downarrow{\scriptstyle \mu} \\ \mu\alpha'.[\alpha']\,\underline{(\mu\beta.[\gamma]\,x)\,u} & \not\simeq_\sigma & \mu\alpha'.[\gamma]\,x \end{array} \tag{1.9}$$

An additional μ-step will be needed on the left (the underlined one), which is not present on the right, to be able to obtain a term equivalent to $\mu\alpha'.[\gamma]\,x$ on the right. Hence, this does not constitute a strong bisimulation diagram. However, completely dropping $\simeq_\rho$ is not possible since it is required to be able to swap renamings. For example, the following identity

$$[\alpha']\,\mu\alpha.[\beta']\,\mu\beta.c \simeq [\beta']\,\mu\beta.[\alpha']\,\mu\alpha.c$$

where $\beta \neq \alpha', \alpha \neq \beta'$, can be deduced using $\simeq_\rho$ twice. This swapping identity (and two others, see Section 6 for details) are necessary to be able to close other strong bisimulation diagrams. Example 5.10 in Section 5 illustrates this point. As it turns out, if one drops $\simeq_\rho$ but retains such swapping equations, just "enough of $\simeq_\rho$" is preserved to obtain our strong bisimulation result.

Contributions. Our contributions may be summarised as follows:

(1) A refinement of $\lambda\mu$, called ΛM-calculus, including explicit substitutions for variables, and explicit replacement for names. The ΛM-calculus is proved to be confluent (Theorem 4.3);
(2) A notion of structural equivalence $\simeq$ for ΛM that is a strong bisimulation with respect to meaningful computation on the set of plain normal forms. (Theorem 5.9).
(3) A precise correspondence result between our bisimulation $\simeq$ on ΛM-objects and Laurent's original σ-equivalence on $\lambda\mu$-objects.

This paper is an extended and revised version of [KBV20].

Structure of the paper. After some preliminaries on notation introduced in Section 2, Section 3 and Section 4 present $\lambda\mu$ and ΛM, respectively. Section 5 discusses the difficulties in formulating a strong bisimulation and presents the proposed solution. Section 6 addresses the correspondence proof between our bisimulation $\simeq$ on ΛM-objects and Laurent's original σ-equivalence on $\lambda\mu$-objects. Finally, Section 7 concludes and describes related work. Most proofs are relegated to the Appendix.

2. Some Basic Preliminary Notions

We start this section by some generic notations. Let $\mathcal{R}$ be any reduction relation on a set of elements $\mathcal{O}$. We write $\twoheadrightarrow_{\mathcal{R}}$ for the reflexive-transitive closure of $\rightarrow_{\mathcal{R}}$. If $\mathcal{S}$ is another reduction relation on $\mathcal{O}$, we use $\rightarrow_{\mathcal{R},\mathcal{S}}$ to denote $\rightarrow_{\mathcal{R}} \cup \rightarrow_{\mathcal{S}}$. We say there is an $\mathcal{R}$-reduction sequence starting at $t_0 \in \mathcal{O}$ if there exists $t_1 \ldots, t_n \in \mathcal{O}$ with $n \geq 0$, such that $t_0 \rightarrow_{\mathcal{R}} t_1$, $t_1 \rightarrow_{\mathcal{R}} t_2$, …, $t_{n-1} \rightarrow_{\mathcal{R}} t_n$. We occasionally refer to $\mathcal{R}$-reduction as $\mathcal{R}$-computation. An element $t \in \mathcal{O}$ enjoys the **$\mathcal{R}$-diamond property** iff for every $u \neq v \in \mathcal{O}$ such that $t \rightarrow_{\mathcal{R}} u$ and $t \rightarrow_{\mathcal{R}} v$, there exists t' such that $u \rightarrow_{\mathcal{R}} t'$ and $v \rightarrow_{\mathcal{R}} t'$. An element $t \in \mathcal{O}$ is said to be **$\mathcal{R}$-confluent** iff for every $u, v \in \mathcal{O}$ such that $t \twoheadrightarrow_{\mathcal{R}} u$ and $t \twoheadrightarrow_{\mathcal{R}} v$, there exists t' such that $u \twoheadrightarrow_{\mathcal{R}} t'$ and $v \twoheadrightarrow_{\mathcal{R}} t'$. A reduction relation $\mathcal{R}$ has the **diamond property** iff every $t \in \mathcal{O}$ has the $\mathcal{R}$-diamond property. A reduction relation $\mathcal{R}$ is **confluent** iff every $t \in \mathcal{O}$ is $\mathcal{R}$-confluent. If $\mathcal{R}$ has the diamond property, then $\mathcal{R}$ is confluent [BN98]. An element $t \in \mathcal{O}$ is said to be **$\mathcal{R}$-terminating** iff there is no infinite $\mathcal{R}$-reduction sequence starting at t. A reduction relation $\mathcal{R}$ is **terminating** iff every $t \in \mathcal{O}$ is $\mathcal{R}$-terminating. An element $t \in \mathcal{O}$ is said to be in **$\mathcal{R}$-normal form** iff there is no t' such that $t \rightarrow_{\mathcal{R}} t'$. We use $\mathcal{NF}_{\mathcal{R}}$ to denote all the normal forms of $\mathcal{R}$, *i.e.* the set of all the elements in $\mathcal{O}$ which are in $\mathcal{R}$-normal form. Given $t \in \mathcal{O}$, we say that u is **an $\mathcal{R}$-normal form of** t if $t \twoheadrightarrow_{\mathcal{R}} u$ and u is in $\mathcal{R}$-normal form. We denote by $\mathsf{nf}_{\mathcal{R}}(t)$ the set of all $\mathcal{R}$-normal forms of t; this set is always a singleton when $\mathcal{R}$ is confluent and terminating.

3. The $\lambda\mu$-Calculus

In this section we introduce the untyped $\lambda\mu$-calculus.

3.1. The Untyped $\lambda\mu$-Calculus.

Given a countably infinite set of variables $\mathbb{V}$ $(x, y, \ldots)$ and names $\mathbb{N}$ $(\alpha, \beta, \ldots)$, the set of **objects** $\mathbb{O}_{\lambda\mu}$, **terms** $\mathbb{T}_{\lambda\mu}$, **commands** $\mathbb{C}_{\lambda\mu}$ and **contexts** of the $\lambda\mu$-calculus are defined by means of the following grammar:

$$
\begin{array}{rrcl}
\textbf{(Objects)} & o & ::= & t \mid c \\
\textbf{(Terms)} & t & ::= & x \mid t\,t \mid \lambda x.t \mid \mu\alpha.c \\
\textbf{(Commands)} & c & ::= & [\alpha]\,t \\
\textbf{(Contexts)} & \mathtt{O} & ::= & \mathtt{T} \mid \mathtt{C} \\
\textbf{(Term Contexts)} & \mathtt{T} & ::= & \Box \mid \mathtt{T}\,t \mid t\,\mathtt{T} \mid \lambda x.\mathtt{T} \mid \mu\alpha.\mathtt{C} \\
\textbf{(Command Contexts)} & \mathtt{C} & ::= & \boxdot \mid [\alpha]\,\mathtt{T}
\end{array}
$$

The grammar extends the terms of the λ-calculus with two new constructors: **commands** $[\alpha]\,t$ and **μ-abstractions** $\mu\alpha.c$. The combination of a command and a μ-abstraction will be coined **explicit renaming**, as in $[\alpha]\,\mu\beta.c$. The term $(\ldots((t\,u_1)\,u_2)\ldots)\,u_n$ abbreviates as $t\,u_1\,u_2\ldots u_n$ or $t\,\vec{u}$ when n is clear from the context. Regarding contexts, there are two holes $\Box$ and $\boxdot$ of sort **term** (t) and **command** (c) respectively. We write $\mathtt{O}\langle o\rangle$ to denote the replacement of the hole $\Box$ (resp. $\boxdot$) by a term (resp. by a command). We often decorate contexts or functions over expressions with one of the sorts t and c to be more clear. For example, $\mathtt{O}_\mathsf{t}$ is a context $\mathtt{O}$ with a hole of sort **term**. The subscript is omitted if it is clear from the context.

Free and **bound variables** of objects are defined as expected, in particular $\mathsf{fv}(\mu\alpha.c) \triangleq \mathsf{fv}(c)$ and $\mathsf{fv}([\alpha]\,t) \triangleq \mathsf{fv}(t)$. **Free** and **bound names** are defined as follows:

$$\begin{array}{rcl@{\qquad\qquad}rcl}
\mathsf{fn}(x) & \triangleq & \emptyset & \mathsf{bn}(x) & \triangleq & \emptyset \\
\mathsf{fn}(t\,u) & \triangleq & \mathsf{fn}(t) \cup \mathsf{fn}(u) & \mathsf{bn}(t\,u) & \triangleq & \mathsf{bn}(t) \cup \mathsf{bn}(u) \\
\mathsf{fn}(\lambda x.t) & \triangleq & \mathsf{fn}(t) & \mathsf{bn}(\lambda x.t) & \triangleq & \mathsf{bn}(t) \\
\mathsf{fn}(\mu\alpha.c) & \triangleq & \mathsf{fn}(c) \setminus \{\alpha\} & \mathsf{bn}(\mu\alpha.c) & \triangleq & \mathsf{bn}(c) \cup \{\alpha\} \\
\mathsf{fn}([\alpha]\,t) & \triangleq & \mathsf{fn}(t) \cup \{\alpha\} & \mathsf{bn}([\alpha]\,t) & \triangleq & \mathsf{bn}(t)
\end{array}$$

We use $\mathsf{fv}_x(o)$ and $\mathsf{fn}_\alpha(o)$ to denote the number of free occurrences of the variable x and the name α in the object o respectively. Additionally, we write $x \notin o$ ($\alpha \notin o$) when $x \notin \mathsf{fv}(o) \cup \mathsf{bv}(o)$ (respectively $\alpha \notin \mathsf{fn}(o) \cup \mathsf{bn}(o)$). This notion is naturally extended to contexts.

We work with the standard notion of α-conversion, *i.e.* renaming of bound variables and names, thus for example $[\delta]\,(\mu\alpha.[\alpha]\,\lambda x.x)\,z =_\alpha [\delta]\,(\mu\beta.[\beta]\,\lambda y.y)\,z$. In particular, when using two different symbols to denote bound variables or names, we assume that they are different without explicitly mentioning it.

Application of the **implicit substitution** $\{x \setminus u\}$ to the object o, written $o\{x \setminus u\}$, may require α-conversion in order to avoid capture of free variables/names, and it is defined as expected.

Application of the **implicit replacement** $\{\!\{\alpha \setminus^{\alpha'} u\}\!\}$ to an object o, written $o\{\!\{\alpha \setminus^{\alpha'} u\}\!\}$, passes the term u as an argument to any sub-command of o of the form $[\alpha]\,t$ and changes the name of α to α'. This operation is also defined modulo α-conversion in order to avoid the capture of free variables/names. Formally:

$$\begin{array}{rcll}
x\{\!\{\alpha \setminus^{\alpha'} u\}\!\} & \triangleq & x & \\
(t\,v)\{\!\{\alpha \setminus^{\alpha'} u\}\!\} & \triangleq & t\{\!\{\alpha \setminus^{\alpha'} u\}\!\}\,v\{\!\{\alpha \setminus^{\alpha'} u\}\!\} & \\
(\lambda x.t)\{\!\{\alpha \setminus^{\alpha'} u\}\!\} & \triangleq & \lambda x.t\{\!\{\alpha \setminus^{\alpha'} u\}\!\} & x \notin u \\
(\mu\beta.c)\{\!\{\alpha \setminus^{\alpha'} u\}\!\} & \triangleq & \mu\beta.c\{\!\{\alpha \setminus^{\alpha'} u\}\!\} & \beta \notin u, \beta \neq \alpha' \\
([\alpha]\,c)\{\!\{\alpha \setminus^{\alpha'} u\}\!\} & \triangleq & [\alpha']\,(c\{\!\{\alpha \setminus^{\alpha'} u\}\!\}\,u) & \\
([\beta]\,c)\{\!\{\alpha \setminus^{\alpha'} u\}\!\} & \triangleq & [\beta]\,c\{\!\{\alpha \setminus^{\alpha'} u\}\!\} & \beta \neq \alpha
\end{array}$$

For example, if $I = \lambda w.w$, then

$$\begin{array}{rcl}
((\mu\alpha.[\alpha]\,x)\,(\lambda z.z\,x))\{x \setminus I\} & = & (\mu\alpha.[\alpha]\,I)\,(\lambda z.z\,I) \\
([\alpha]\,x\,(\mu\beta.[\alpha]\,y))\{\!\{\alpha \setminus^{\alpha'} I\}\!\} & = & [\alpha']\,x\,(\mu\beta.[\alpha']\,y\,I)\,I
\end{array}$$

Parigot's original formulation of $\lambda\mu$-calculus [Par92, Par93] uses a binary replacement operation $c\{\!\{\alpha \setminus u\}\!\}$ rather than the ternary one we introduced above. Details on our choice of notation, which are related to explicit replacements, are developed in Section 4.2.

The **one-step reduction relation** $\rightarrow_{\lambda\mu}$ is given by the closure by *all* contexts $\mathtt{O}_\mathsf{t}$ of the following rewriting rules β and μ, *i.e.* $\rightarrow_{\lambda\mu} \triangleq \mathtt{O}_\mathsf{t}\langle \mapsto_\beta \cup \mapsto_\mu \rangle$:

$$\begin{array}{rcl}
(\lambda x.t)\,u & \mapsto_\beta & t\{x \setminus u\} \\
(\mu\alpha.c)\,u & \mapsto_\mu & \mu\alpha'.c\{\!\{\alpha \setminus^{\alpha'} u\}\!\}
\end{array}$$

Given $X \in \{\beta, \mu\}$, we define an X**-redex** to be a term having the form of the left-hand side of the rule $\mapsto_X$. A similar notion will be used for all the rewriting rules used in this paper. It is worth noticing that Parigot's [Par92] μ-rule of the $\lambda\mu$-calculus relies on a binary implicit replacement operation $\{\!\{\alpha \setminus u\}\!\}$ assigning $[\alpha]\,(t\{\!\{\alpha \setminus u\}\!\})\,u$ to each sub-expression of the form $[\alpha]\,t$ (thus not changing the name of the command). We remark that $\mu\alpha.c\{\!\{\alpha \setminus u\}\!\} =_\alpha \mu\alpha'.c\{\!\{\alpha \setminus^{\alpha'} u\}\!\}$; thus *e.g.* $\mu\alpha.([\alpha]\,x)\{\!\{\alpha \setminus u\}\!\} = \mu\alpha.[\alpha]\,x\,u =_\alpha \mu\gamma.[\gamma]\,x\,u = \mu\gamma.([\alpha]\,x)\{\!\{\alpha \setminus^{\gamma} u\}\!\}$. We adopt here the ternary presentation [KV19] of the implicit replacement operator, because it naturally extends to that of the ΛM-calculus in Section 4.

Various control operators can be expressed in the $\lambda\mu$-calculus [dG94, Lau03]. A typical example is the control operator **call-cc** [Gri90], specified by the term $\lambda x.\mu\alpha.[\alpha]\,x\,(\lambda y.\mu\delta.[\alpha]\,y)$.

3.2. The notion of σ-equivalence for $\lambda\mu$-terms.

As in λ-calculus, structural equivalence for the $\lambda\mu$-calculus captures inessential permutation of redexes, but this time also involving the control constructs.

Definition 3.1. Laurent's notion of σ-equivalence for $\lambda\mu$-objects [Lau03] (written here also $\simeq_\sigma$) is depicted in Figure 3, where $\{\beta \setminus \alpha\}$ denotes the **implicit renaming** of all the free occurrences of the name β by α (a formal definition is given in Section 4.4).

$$
\begin{array}{rcll}
(\lambda y.\lambda x.t)\,v & \simeq_{\sigma_1} & \lambda x.(\lambda y.t)\,v & x \notin v \\
(\lambda x.t\,v)\,u & \simeq_{\sigma_2} & (\lambda x.t)\,u\,v & x \notin v \\
(\lambda x.\mu\alpha.[\beta]\,u)\,w & \simeq_{\sigma_3} & \mu\alpha.[\beta]\,(\lambda x.u)\,w & \alpha \notin w \\
[\alpha']\,(\mu\alpha.[\beta']\,(\mu\beta.c)\,w)\,v & \simeq_{\sigma_4} & [\beta']\,(\mu\beta.[\alpha']\,(\mu\alpha.c)\,v)\,w & \alpha \notin w, \beta \notin v, \beta \neq \alpha', \alpha \neq \beta' \\
[\alpha']\,(\mu\alpha.[\beta']\,\lambda x.\mu\beta.c)\,v & \simeq_{\sigma_5} & [\beta']\,\lambda x.\mu\beta.[\alpha']\,(\mu\alpha.c)\,v & x \notin v, \beta \notin v, \beta \neq \alpha', \alpha \neq \beta' \\
[\alpha']\,\lambda x.\mu\alpha.[\beta']\,\lambda y.\mu\beta.c & \simeq_{\sigma_6} & [\beta']\,\lambda y.\mu\beta.[\alpha']\,\lambda x.\mu\alpha.c & \beta \neq \alpha', \alpha \neq \beta' \\
\mu\alpha.[\alpha]\,v & \simeq_{\sigma_7} & v & \alpha \notin v \\
[\alpha]\,\mu\beta.c & \simeq_{\sigma_8} & c\{\beta \setminus \alpha\} &
\end{array}
$$

Figure 3: σ-equivalence for $\lambda\mu$-objects

The first two equations are exactly those of Regnier (hence $\simeq_\sigma$ on $\lambda\mu$-terms strictly extends $\simeq_\sigma$ on λ-terms); the remaining ones involve μ-abstractions. It is worth noticing that our equations $\simeq_{\sigma_7}$ and $\simeq_{\sigma_8}$ are called, respectively, $\simeq_\theta$ and $\simeq_\rho$ in [Lau03].

Laurent proved properties for $\simeq_\sigma$ on $\lambda\mu$-terms similar to those of Regnier for $\simeq_\sigma$ on λ-terms. More precisely, $u \simeq_\sigma v$ implies that u is normalisable (resp. is head normalisable, strongly normalisable) iff v is normalisable (resp. is head normalisable, strongly normalisable) [Lau03, Proposition 35]. Based on Girard's encoding of classical into intuitionistic logic [Gir91], he also proved that the translation of the left and right-hand sides of the equations of $\simeq_\sigma$, in a typed setting, yield structurally equivalent (polarised) proof-nets [Lau03, Theorem 41]. These results are non-trivial because the left and right-hand side of the equations in Figure 3 do not have the same β and μ redexes. For example, $(\mu\alpha.[\alpha]\,x)\,y$ and $x\,y$ are related by equation σ_7, however the former has a μ-redex (more precisely it has a *linear* μ-redex) and the latter has none. Indeed, $\simeq_\sigma$ is not a strong bisimulation with respect to $\lambda\mu$-reduction, as mentioned in the introduction (*cf.* the terms in (1.5)):

$$
\begin{array}{ccc}
(\mu\alpha.[\alpha]\,x)\,y & \simeq_{\sigma_8} & x\,y \\
{\scriptstyle\mu}\downarrow & & \not\downarrow{\scriptstyle\mu} \\
\mu\alpha.[\alpha]\,x\,y & \simeq_{\sigma_8} & x\,y
\end{array}
\tag{3.1}
$$

The above diagram shows, moreover, that an analogue of Theorem 1.1 does not hold for $\lambda\mu$. There are other examples illustrating that $\simeq_\sigma$ is not a strong bisimulation (*cf.* Section 5). It seems natural to wonder whether, just like in the intuitionistic case, a more refined notion of $\lambda\mu$-reduction could change this state of affairs; a challenge we take up in this paper.

4. The ΛM-calculus

As a first step towards the definition of an adequate strongly bisimilar structural equivalence for the $\lambda\mu$-calculus, we extend its syntax and operational semantics to a term calculus with explicit operators for substitution and replacement.

4.1. Terms for ΛM.

We consider again a countably infinite set of **variables** $\mathbb{V}$ $(x, y, \ldots)$ and **names** $\mathbb{N}$ $(\alpha, \beta, \ldots)$. The set of **objects** $\mathbb{O}_{\Lambda M}$, **terms** $\mathbb{T}_{\Lambda M}$, **commands** $\mathbb{C}_{\Lambda M}$, **stacks** and **contexts** of the ΛM-calculus are given by the following grammar:

$$
\begin{array}{rrcl}
\textbf{(Objects)} & o & ::= & t \mid c \mid s \\
\textbf{(Terms)} & t & ::= & x \mid t\,t \mid \lambda x.t \mid \mu\alpha.c \mid t[x \setminus t] \\
\textbf{(Commands)} & c & ::= & [\alpha]\,t \mid c[\![\alpha \setminus^{\alpha'} s]\!] \\
\textbf{(Stacks)} & s & ::= & t \mid t \cdot s \\
\textbf{(Contexts)} & \mathtt{O} & ::= & \mathtt{T} \mid \mathtt{C} \mid \mathtt{S} \\
\textbf{(Term Contexts)} & \mathtt{T} & ::= & \Box \mid \mathtt{T}\,t \mid t\,\mathtt{T} \mid \lambda x.\mathtt{T} \mid \mu\alpha.\mathtt{C} \mid \mathtt{T}[x \setminus t] \mid t[x \setminus \mathtt{T}] \\
\textbf{(Command Contexts)} & \mathtt{C} & ::= & \boxdot \mid [\alpha]\,\mathtt{T} \mid \mathtt{C}[\![\alpha \setminus^{\alpha'} s]\!] \mid c[\![\alpha \setminus^{\alpha'} \mathtt{S}]\!] \\
\textbf{(Stack Contexts)} & \mathtt{S} & ::= & \mathtt{T} \mid \mathtt{T} \cdot s \mid t \cdot \mathtt{S} \\
\textbf{(Substitution Contexts)} & \mathtt{L} & ::= & \Box \mid \mathtt{L}[x \setminus t] \\
\textbf{(Repl./Ren. Contexts)} & \mathtt{R} & ::= & \boxdot \mid \mathtt{R}[\![\alpha \setminus^{\alpha'} s]\!] \mid [\beta]\,\mu\alpha.\mathtt{R}
\end{array}
$$

Terms are those of the $\lambda\mu$-calculus enriched with **explicit substitutions (ES)** of the form $[x \setminus u]$. The subterm u in a term of the form $t\,u$ (resp. the ES $t[x \setminus u]$) is called the **argument** of the application (resp. substitution). Commands are enriched with **explicit replacements** of the form $[\![\alpha \setminus^{\alpha'} s]\!]$ (where the stack s is to be considered as list of arguments, as *e.g.* in [Her94]). Notice that stacks inside explicit replacements are required to be *non-empty*.

Stacks can be concatenated as expected (denoted $s \cdot s'$ by abuse of notation): if $s = t_0 \cdot \ldots \cdot t_n$, then $s \cdot s' \triangleq t_0 \cdot \ldots \cdot t_n \cdot s'$; where $\_ \cdot \_$ is right associative. Given a term u, we use the abbreviation $u :: s$ for the term resulting from the application of u to all the terms of the stack s, *i.e.* if $s = t_0 \cdot \ldots \cdot t_n$, then $u :: s \triangleq u\,t_0 \ldots t_n$. Recall that application is left associative, so that this operation also is; hence $u :: s :: s'$ means $(u :: s) :: s'$. The use of stacks in the new calculus is motivated with the forthcoming example just after the definition of the implicit replacement in Section 4.3.

Free and **bound variables** of ΛM-objects are defined as expected, having the new explicit operators binding symbols: *i.e.* $\mathsf{fv}(t[x \setminus u]) \triangleq (\mathsf{fv}(t) \setminus \{x\}) \cup \mathsf{fv}(u)$ and $\mathsf{bv}(t) =$

$\mathsf{bv}(t) \cup \mathsf{bv}(u) \cup \{x\}$. Concerning **free** and **bound names** of ΛM-objects, we remark in particular that the occurrences of α' in the explicit replacements $c[\![\alpha \setminus^{\alpha'} s]\!]$ are not bound:

$$\mathsf{fn}(c[\![\alpha \setminus^{\alpha'} s]\!]) \triangleq (\mathsf{fn}(c) \setminus \{\alpha\}) \cup \mathsf{fn}(s) \cup \{\alpha'\} \qquad \mathsf{bn}(c[\![\alpha \setminus^{\alpha'} s]\!]) \triangleq \mathsf{bn}(c) \cup \mathsf{bn}(s) \cup \{\alpha\}$$

We work, as usual, modulo α-conversion so that bound variables and names can be renamed. Thus *e.g.* $x[x \setminus u] =_\alpha y[y \setminus u]$, and $([\gamma]\, x)[\![\gamma \setminus^\alpha u]\!] =_\alpha ([\beta]\, x)[\![\beta \setminus^\alpha u]\!]$. In particular, we assume by α-conversion that $x \notin \mathsf{fv}(u)$ in $t[x \setminus u]$, and $\alpha \notin \mathsf{fn}(s)$ in $c[\![\alpha \setminus^{\alpha'} s]\!]$.

The notions of free and bound variables and names are extended to contexts by defining $\mathsf{fv}(\Box) = \mathsf{fv}(\boxdot) = \mathsf{fn}(\Box) = \mathsf{fn}(\boxdot) = \emptyset$. Then *e.g.* x is bound in $\lambda x.\Box$, $(\lambda x.x)\,\Box$, and α is bound in $\boxdot[\![\alpha \setminus^{\alpha'} s]\!]$. Bound names whose scope includes a hole $\Box$ or $\boxdot$ cannot be α-renamed. An object o is **free for a context** $\mathtt{O}$, written $\mathsf{fc}(o, \mathtt{O})$, if $\mathsf{fv}(o)$ are not captured by binders of $\mathtt{O}$ in $\mathtt{O}\langle o\rangle$. Thus *e.g.* $\mathsf{fc}(z\,y, \lambda x.\Box[x' \setminus w])$ and $\mathsf{fc}(x, (\lambda x.x)\,\Box)$ hold but $\mathsf{fc}(x\,y, \lambda x.\Box)$ does not hold. This notion is naturally extended to sets of objects, *i.e.* $\mathsf{fc}(\mathcal{S}, \mathtt{O})$ iff $\mathsf{fc}(o, \mathtt{O})$ holds for every $o \in \mathcal{S}$.

4.2. **On Choice of Notation for ΛM.** The decomposition (1.6) of Parigot's $\lambda\mu$-calculus [Par92, Par93] mentioned in the introduction, is based on Andou's formalization [And03]. Alternatively, adopting an explicit formulation of Parigot's original replacement operation, results in:

$$\begin{array}{rcl} (\mu\alpha.c)\, u & \mapsto & \mu\alpha.c[\![\alpha \setminus u]\!] \\ c[\![\alpha \setminus u]\!] & \mapsto & c\{\!\{\alpha \setminus u\}\!\} \end{array} \tag{4.1}$$

This alternative has the advantage of being relatively simple. However, it is not without its subtleties. Most notable is determining the status of names. Consider an expression such as $\mu\alpha.c[\![\alpha \setminus u]\!][\![\alpha \setminus v]\!]$, resulting from reducing $(\mu\alpha.c)\,u\,v$. One might understand that names are bound by multiple binders. Occurrences of α in c would thus be bound by three operators: the outermost $\mu\alpha$ and the two explicit replacements $[\![\alpha \setminus u]\!]$ and $[\![\alpha \setminus v]\!]$. Alternatively, the occurrences of α in $[\![\alpha \setminus u]\!]$ and $[\![\alpha \setminus v]\!]$ could be understood as free. In this case, the outermost $\mu\alpha$ binds all free occurrences of α in c and the two occurrences of α in $[\![\alpha \setminus u]\!]$ and $[\![\alpha \setminus v]\!]$. Beyond settling for one of these two approaches, there is the additional issue that firing $[\![\alpha \setminus v]\!]$ actually doesn't affect α in c at all, for otherwise the ordering of u and v should be confused. The notion of scope is lost, and hence the ordering between $[\![\alpha \setminus u]\!]$ and $[\![\alpha \setminus v]\!]$. Presentation (1.6) is somewhat heavier but crisper in terms of meaning. The previously mentioned term would be recast as $\mu\alpha''.c[\![\alpha \setminus^{\alpha'} u]\!][\![\alpha' \setminus^{\alpha''} v]\!]$. Here α in c is bound to just one operator, namely $c[\![\alpha \setminus^{\alpha'} u]\!]$. Moreover, the dependency between u and v is now readily apparent: α' is bound by $[\![\alpha' \setminus^{\alpha''} v]\!]$ and α'' is bound by $\mu\alpha''$. In particular, $\mu\alpha''$ does not bind any name in c. Presentation (1.7) which we recall below and is the one used in this paper:

$$\begin{array}{rcl} (\mu\alpha.c)[x_1 \setminus v_1] \ldots [x_n \setminus v_n]\, u & \mapsto_{\mathtt{dM}} & (\mu\alpha'.c[\![\alpha \setminus^{\alpha'} u]\!])[x_1 \setminus v_1] \ldots [x_n \setminus v_n] \\ c[\![\alpha \setminus^{\alpha'} u]\!] & \mapsto_{\mathtt{R}} & c\{\!\{\alpha \setminus^{\alpha'} u\}\!\} \end{array}$$

has an additional benefit that we shall not get to exploit here but that may be done so in a continuation of this work (which originally sparked it, in fact). It has to do with *single replacement*, where one would like to perform replacement of one occurrence of $[\alpha]\,t$ at a time. Consider a term such as $t_0 = (\ldots [\alpha]\,x \ldots [\alpha]\,y \ldots)[\![\alpha \setminus u]\!]$. We could first pass the argument u to x, yielding $t_1 = (\ldots [\alpha]\,xu \ldots [\alpha]\,y \ldots) \ldots$, and then to y, yielding $t_2 = (\ldots [\alpha]\,xu \ldots [\alpha]\,yu \ldots) \ldots$. Note that the partially replaced term t_1 must be represented somehow, and in the simplified syntax we would write $(\ldots [\alpha]\,xu \ldots [\alpha]\,y \ldots)[\![\alpha \setminus u]\!]$ which does not make any sense. Indeed, the first argument named α in t_1 has already been replaced while the second one is still waiting for an argument, a fact not reflected in the syntax. It is exactly in this framework that the ternary notion of replacement inherited from Andou [And03] makes sense. Our example now reads $t_0 = (\ldots [\alpha]\, x \ldots [\alpha]\, y \ldots)[\![\alpha \setminus^{\alpha'} u]\!]$ which first reduces to $t_1 = (\ldots [\alpha']\, xu \ldots [\alpha]\, y \ldots)[\![\alpha \setminus^{\alpha'} u]\!]$, then to $(\ldots [\alpha']\, xu \ldots [\alpha']\, yu \ldots)[\![\alpha \setminus^{\alpha'} u]\!]$.

4.3. **Substitution, Renaming and Replacement in ΛM.** The **application of the implicit substitution** $\{x \setminus u\}$ to an ΛM-object o is defined as the natural extension of that of the $\lambda\mu$-calculus (Section 3.1). We now detail the applications of implicit replacements and renamings, which are more subtle. The **application of the implicit replacement** $\{\!\{\alpha \setminus^{\alpha'} s\}\!\}$ to an ΛM-object, is defined as the following capture-avoiding operation (recall that by α-conversion $\alpha \notin \mathsf{fn}(s)$ and $\alpha \neq \alpha'$):

$$
\begin{array}{rcll}
x\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \triangleq & x & \\
(t\,u)\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \triangleq & t\{\!\{\alpha \setminus^{\alpha'} s\}\!\}\, u\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \\
(\lambda x.t)\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \triangleq & \lambda x.t\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & x \notin s \\
(\mu\beta.c)\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \triangleq & \mu\beta.c\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \beta \notin s, \beta \neq \alpha' \\
(t[x \setminus u])\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \triangleq & (t\{\!\{\alpha \setminus^{\alpha'} s\}\!\})[x \setminus u\{\!\{\alpha \setminus^{\alpha'} s\}\!\}] & x \notin s \\
([\alpha]\, t)\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \triangleq & [\alpha']\, (t\{\!\{\alpha \setminus^{\alpha'} s\}\!\}) :: s & \\
([\beta]\, t)\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \triangleq & [\beta]\, t\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \alpha \neq \beta \\
(c[\![\gamma \setminus^{\alpha} s']\!])\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \triangleq & (c\{\!\{\alpha \setminus^{\alpha'} s\}\!\})[\![\gamma \setminus^{\alpha'} s'\{\!\{\alpha \setminus^{\alpha'} s\}\!\} \cdot s]\!] & \gamma \notin s, \gamma \neq \alpha' \\
(c[\![\gamma \setminus^{\beta} s']\!])\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \triangleq & (c\{\!\{\alpha \setminus^{\alpha'} s\}\!\})[\![\gamma \setminus^{\beta} s'\{\!\{\alpha \setminus^{\alpha'} s\}\!\}]\!] & \alpha \neq \beta, \gamma \notin s, \gamma \neq \alpha' \\
(t \cdot s')\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \triangleq & t\{\!\{\alpha \setminus^{\alpha'} s\}\!\} \cdot s'\{\!\{\alpha \setminus^{\alpha'} s\}\!\} &
\end{array}
$$

Most of the cases in the definition above are straightforward, we only comment on the interesting ones. When the implicit replacement affects an explicit replacement, *i.e.* in the case $(c[\![\gamma \setminus^{\alpha} s']\!])\{\!\{\alpha \setminus^{\alpha'} s\}\!\}$, the explicit replacement is *blocking* the implicit replacement operation over γ. This means that γ and α denote the same command, but the arguments of α must not be passed to γ yet. This is why the resulting explicit replacement will accumulate all these arguments in a *stack*, which explains the need for this data structure inside explicit replacements. Examples of these operations are $([\alpha]\, x)\{\!\{\alpha \setminus^{\gamma} y_0 \cdot y_1\}\!\} = [\gamma]\, x\, y_0\, y_1$, and $(([\alpha]\, x)[\![\beta \setminus^{\alpha} z_0]\!])\{\!\{\alpha \setminus^{\gamma} y_0\}\!\} = ([\gamma]\, x\, y_0)[\![\beta \setminus^{\gamma} z_0 \cdot y_0]\!]$.

The **application of the implicit renaming** $\{\alpha \setminus \beta\}$ to an ΛM-object is defined as:

$$
\begin{array}{rcll}
x\{\alpha \setminus \beta\} & \triangleq & x & \\
(t\,u)\{\alpha \setminus \beta\} & \triangleq & t\{\alpha \setminus \beta\}\, u\{\alpha \setminus \beta\} & \\
(\lambda x.t)\{\alpha \setminus \beta\} & \triangleq & \lambda x.t\{\alpha \setminus \beta\} & \\
(\mu\gamma.c)\{\alpha \setminus \beta\} & \triangleq & \mu\gamma.c\{\alpha \setminus \beta\} & \gamma \neq \beta \\
(t[x \setminus u])\{\alpha \setminus \beta\} & \triangleq & (t\{\alpha \setminus \beta\})[x \setminus u\{\alpha \setminus \beta\}] & \\
([\alpha]\, t)\{\alpha \setminus \beta\} & \triangleq & [\beta]\, t\{\alpha \setminus \beta\} & \\
([\delta]\, t)\{\alpha \setminus \beta\} & \triangleq & [\delta]\, t\{\alpha \setminus \beta\} & \alpha \neq \delta \\
(c[\![\gamma \setminus^{\alpha} s]\!])\{\alpha \setminus \beta\} & \triangleq & (c\{\alpha \setminus \beta\})[\![\gamma \setminus^{\beta} s\{\alpha \setminus \beta\}]\!] & \gamma \neq \beta \\
(c[\![\gamma \setminus^{\delta} s]\!])\{\alpha \setminus \beta\} & \triangleq & (c\{\alpha \setminus \beta\})[\![\gamma \setminus^{\delta} s\{\alpha \setminus \beta\}]\!] & \alpha \neq \delta, \gamma \neq \beta \\
(t \cdot s)\{\alpha \setminus \beta\} & \triangleq & t\{\alpha \setminus \beta\} \cdot s\{\alpha \setminus \beta\} &
\end{array}
$$

The three operations $\{x \setminus u\}$, $\{\!\{\alpha \setminus^{\alpha'} s\}\!\}$ and $\{\alpha \setminus \beta\}$ are extended to contexts as expected. The table below summarises the notions of implicit and explicit operations introduced above:

$t\{x \setminus u\}$ $t[x \setminus u]$	Implicit Substitution Explicit Substitution	Substitute a term for a variable
$c\{\{\alpha \setminus^{\alpha'} s\}\}$ $c[[\alpha \setminus^{\alpha'} s]]$	Implicit Replacement Explicit Replacement	Forwards arguments to named terms
$c\{\alpha \setminus \beta\}$	Implicit Renaming	Substitute a name for a name

4.4. **Reduction Semantics of ΛM.** The reduction semantics for ΛM will be presented in terms of reduction rules. With an eye placed on the upcoming notion of structural equivalence $\simeq$, we will classify these rules as performing mere reshuffling of symbols or performing more elaborate work. This classification is motivated by the multiplicative and exponential nature of the different redexes of terms of ΛM as discussed in the introduction.

The first two reduction rules for ΛM arise from the simple decomposition of β-reduction:

$$\begin{array}{rcl} \mathtt{L}\langle \lambda x.t\rangle\, u & \mapsto_{\mathtt{dB}} & \mathtt{L}\langle t[x \setminus u]\rangle \\ t[x \setminus u] & \mapsto_{\mathtt{S}} & t\{x \setminus u\} \end{array}$$

where $\mapsto_{\mathtt{dB}}$ is constrained by the condition $\mathtt{fc}(u, \mathtt{L})$. These have already been studied in the literature, where, as discussed in the introduction, they suffice to be able to state and prove a strong bisimulation result for the intuitionistic case (*cf.* Theorem 1.2). As mentioned there, the first is a simple reshuffling of symbols, which we thus consider to be *plain* computation, but the second is not. It may substitute u deep within a term or possibly even erase u, involving the use of exponential cuts in its proof-net semantics and hence considered *meaningful* computation. Note that dB operates *at a distance* [AK12a], in the sense that an abstraction and its argument may be separated by an arbitrary number of explicit substitutions.

The next reduction rules we consider arise from a more subtle decomposition of μ-reduction. The third reduction rule for ΛM is:

$$\mathtt{L}\langle \mu\alpha.c\rangle\, u \;\mapsto_{\mathtt{dM}}\; \mathtt{L}\langle \mu\alpha'.c[[\alpha \setminus^{\alpha'} u]]\rangle$$

subject to the constraint $\mathtt{fc}(u, \mathtt{L})$, and α' is a fresh name (*i.e.* $\alpha' \neq \alpha$, $\alpha' \notin c$ and $\alpha' \notin u$). This rule is similar in nature to dB in the sense that it fires a μ-redex and may be seen to reshuffle symbols. In particular no replacement actually takes place since it introduces an explicit replacement. Rule dM is therefore also considered to perform *plain* computation. Note that dM operates at a distance too. With the introduction of dM our notation for explicit replacement can now be justified. Indeed, following Parigot [Par92], one might be tempted to rephrase the reduct of dM with a binary constructor, writing $\mathtt{L}\langle \mu\alpha.c[[\alpha \setminus u]]\rangle$ on the right-hand side of the rule dM. This would be incorrect since all free occurrences of α in c are bound by the α in $c[[\alpha \setminus u]]$ which renders the role of "$\mu\alpha$" meaningless.

We have not yet finished introducing the reduction rules for ΛM. All that is missing is a means to execute the explicit replacement introduced by dM. The natural candidate for executing replacement would be to have just one reduction rule, namely: $c[[\alpha \setminus^{\alpha'} s]] \mapsto_{\mathtt{R}} c\{\{\alpha \setminus^{\alpha'} s\}\}$. However, this is too coarse grained to be able to obtain our strong bisimulation result (*cf.* Section 5) and therefore explicit replacement will be implemented not by one, but rather by multiple reduction rules. In particular, these resulting reduction rules can be easily categorised into *plain* and *meaningful* behavior, according to the following criterion: first, whether the bound name α in the command c occurring in the left-hand side of rule

R is linear; and, second, if this single occurrence of α in c may be erased or duplicated by reduction or not. We will next present these rules gradually.

The first of these rules is the case where α does not occur linearly in c and results in the fourth reduction rule for ΛM:

$$c[\![\alpha \setminus^{\alpha'} s]\!] \quad \mapsto_{\mathtt{Rnl}} \quad c\{\!\{\alpha \setminus^{\alpha'} s\}\!\} \quad \mathsf{fn}_\alpha(c) \neq 1$$

We still have to address the case where there is a unique occurrence of α in c. Replacing that unique occurrence is not necessarily an act of mere reshuffling; it depends on where the occurrence of α appears in c. If α appears inside the argument of an application, an explicit substitution or an explicit replacement, then this single occurrence of α could be further erased or duplicated. One might say replacing s performs *hereditarily* meaningful work. Let us make this more precise. If α occurs exactly once in c, then the left-hand side $c[\![\alpha \setminus^{\alpha'} s]\!]$ of the reduction rule R must have one of the following forms:

$$\mathtt{C}\langle[\boldsymbol{\alpha}]\, t\rangle[\![\alpha \setminus^{\alpha'} s]\!]$$
$$\mathtt{C}\langle c'[\![\beta \setminus^{\boldsymbol{\alpha}} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!]$$

where the unique free occurrence of α in c has been highlighted in bold and α does not occur in any of C, t, c', s'. These determine the following two instances of the reduction rule $c[\![\alpha \setminus^{\alpha'} s]\!] \mapsto_{\mathtt{R}} c\{\!\{\alpha \setminus^{\alpha'} s\}\!\}$ with $\mathsf{fn}_\alpha(c) = 1$:

$$\begin{array}{rcll} \mathtt{C}\langle[\alpha]\, t\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \mapsto_{\mathtt{Name}} & \mathtt{C}\langle[\alpha']\, t :: s\rangle & \alpha \notin \mathtt{C}, \alpha \notin t \\ \mathtt{C}\langle c'[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \mapsto_{\mathtt{Comp}} & \mathtt{C}\langle c'[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle & \alpha \notin \mathtt{C}, \alpha \notin c', \alpha \notin s' \end{array}$$

The first rule applies the explicit replacement when finding the (only) occurrence of the name α; while the second rule composes the explicit replacements by concatenating their respective stacks. As mentioned above, we would like to further identify the case when α appears inside the argument of an application, an explicit substitution or an explicit replacement in each of these reduction rules. The ones where it does are called *non-linear* and the ones where it does not are called *linear*. This results in four (disjoint) rules. From now on, we call N and Nnl the linear and non-linear instances of rule Name respectively; similarly we have C and Cnl for Comp:

Rule	**Linear Instance**	**Non-Linear Instance**
Name	N	Nnl
Comp	C	Cnl

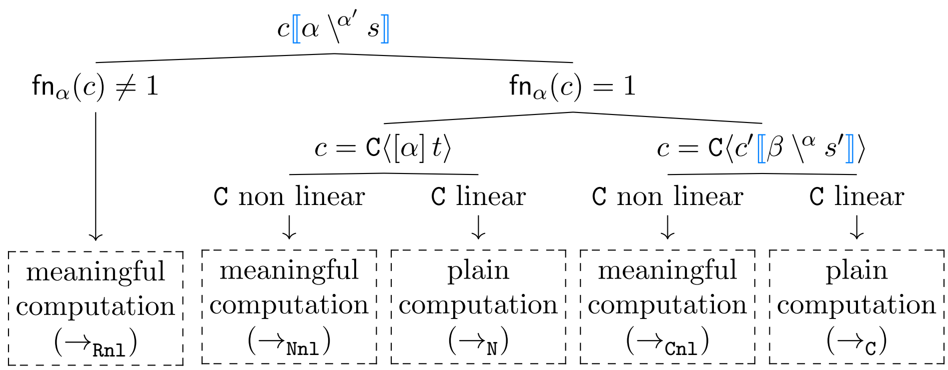
These rules can be formulated with the notion of *linear contexts*, generated by the following grammars:

$$\begin{array}{lrcl} \textbf{(TT Linear Contexts)} & \mathtt{LTT} & ::= & \Box \mid \mathtt{LTT}\, t \mid \lambda x.\mathtt{LTT} \mid \mu\alpha.\mathtt{LCT} \mid \mathtt{LTT}[x \setminus t] \\ \textbf{(TC Linear Contexts)} & \mathtt{LTC} & ::= & \mathtt{LTC}\, t \mid \lambda x.\mathtt{LTC} \mid \mu\alpha.\mathtt{LCC} \mid \mathtt{LTC}[x \setminus t] \\ \textbf{(CC Linear Contexts)} & \mathtt{LCC} & ::= & \boxdot \mid [\alpha]\, \mathtt{LTC} \mid \mathtt{LCC}[\![\alpha \setminus^{\alpha'} s]\!] \\ \textbf{(CT Linear Contexts)} & \mathtt{LCT} & ::= & [\alpha]\, \mathtt{LTT} \mid \mathtt{LCT}[\![\alpha \setminus^{\alpha'} s]\!] \end{array}$$

where each category LXY represents the linear context which takes an object of *sort* Y and returns another of *sort* X. For example, LTC is a linear context taking a command and generating a term. Indeed, notice that the grammar does not allow the hole $\boxdot$ to occur inside a parameter (of an application or an ES). With this definition in place we can, for example, formulate the decomposition of the reduction rule $\mathtt{Name}$ as follows:

$$\begin{array}{rll} \mathtt{LCC}\langle[\alpha]\,t\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \mapsto_{\mathtt{N}} & \mathtt{LCC}\langle[\alpha']\,t :: s\rangle \quad \alpha \notin \mathtt{LCC}, \alpha \notin t \\ \mathtt{C}\langle[\alpha]\,t\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \mapsto_{\mathtt{Nnl}} & \mathtt{C}\langle[\alpha']\,t :: s\rangle \quad \mathtt{C} \text{ non linear}, \alpha \notin \mathtt{C}, \alpha \notin t \end{array}$$

The following diagram summarises which instances of $\mathtt{R}$ are considered plain and which are considered meaningful:



Summarizing our analysis, the full set the reduction rules for ΛM is presented next.

Definition 4.1. Reduction in the **ΛM-calculus** is given by the following reduction rules closed under arbitrary contexts:

$$\begin{array}{rlll} \mathtt{L}\langle\lambda x.t\rangle\, u & \mapsto_{\mathtt{dB}} & \mathtt{L}\langle t[x \setminus u]\rangle & \mathsf{fc}(u, \mathtt{L}) \\ t[x \setminus u] & \mapsto_{\mathtt{S}} & t\{x \setminus u\} & \\ \mathtt{L}\langle\mu\alpha.c\rangle\, u & \mapsto_{\mathtt{dM}} & \mathtt{L}\langle\mu\alpha'.c[\![\alpha \setminus^{\alpha'} u]\!]\rangle & \mathsf{fc}(u, \mathtt{L}), \alpha' \text{ fresh} \\ \mathtt{LCC}\langle[\alpha]\,t\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \mapsto_{\mathtt{N}} & \mathtt{LCC}\langle[\alpha']\,t :: s\rangle & \alpha \notin \mathtt{LCC}, \alpha \notin t \\ \mathtt{LCC}\langle c'[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \mapsto_{\mathtt{C}} & \mathtt{LCC}\langle c'[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle & \alpha \notin \mathtt{LCC}, \alpha \notin c', \alpha \notin s' \\ \mathtt{C}\langle[\alpha]\,t\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \mapsto_{\mathtt{Nnl}} & \mathtt{C}\langle[\alpha']\,t :: s\rangle & \mathtt{C} \text{ non linear}, \alpha \notin \mathtt{C}, \alpha \notin t \\ \mathtt{C}\langle c'[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \mapsto_{\mathtt{Cnl}} & \mathtt{C}\langle c'[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle & \mathtt{C} \text{ non linear}, \alpha \notin \mathtt{C}, \alpha \notin c', \alpha \notin s' \\ c[\![\alpha \setminus^{\alpha'} s]\!] & \mapsto_{\mathtt{Rnl}} & c\{\!\{\alpha \setminus^{\alpha'} s\}\!\} & \mathsf{fn}_\alpha(c) \neq 1 \end{array}$$

Note that $o \to_{\mathtt{R}} o'$ iff o reduces to o' using the reduction obtained from the union of the rules $\mapsto_{\mathtt{N}} \cup \mapsto_{\mathtt{C}} \cup \mapsto_{\mathtt{Nnl}} \cup \mapsto_{\mathtt{Cnl}} \cup \mapsto_{\mathtt{Rnl}}$.

We can then state that the ΛM-calculus refines the $\lambda\mu$-calculus by decomposing β and μ in more atomic steps.

Lemma 4.2. *Let $o \in \mathbb{O}_{\lambda\mu}$. If $o \to_{\lambda\mu} o'$, then $o \twoheadrightarrow_{\Lambda M} o'$.*

Proof. By induction on $o \to_{\lambda\mu} o'$. □

As in the case of the $\lambda\mu$-calculus, the ΛM-calculus is confluent too.

Theorem 4.3. *The reduction relation $\to_{\Lambda M}$ is confluent (CR).*

Proof. The proof uses the interpretation method [CHL96], by projecting the ΛM-calculus into the $\lambda\mu$-calculus. Details in the Appendix A. □

5. A Strong Bisimulation for ΛM

We now introduce our notion of structural equivalence for ΛM, written $\simeq$, breaking down the presentation into two key tools on which we have based our development: plain forms and linear contexts.

Plain Forms. As discussed in Section 1, the initial intuition in defining a strong bisimulation for ΛM arises from the intuitionistic case: Regnier's equivalence σ is not a strong bisimulation, but decomposing the β-rule and taking the $\mathtt{dB}$-normal form of the left and right hand sides of the equations in Figure 1, results in the $\tilde{\sigma}$-equivalence relation terms on λ-terms with explicit substitutions. This relation turns out to be a strong bisimulation with respect to the notion of meaningful computation (the relation $\to_{\mathtt{S}}$ in the case of λ-calculus).

One can identify $\mathtt{dB}$ as performing innocuous or plain computation, a fact that can also be supported by how this step translates as a multiplicative cut in polarized proof-nets [Lau02, Lau03]. Similarly, one can identify $\mathtt{S}$ as performing non-trivial or meaningful work. In the classical case, this leads us to introduce two restrictions of reduction in ΛM (*cf.* Definition 4.1), one called plain and one called meaningful.

Definition 5.1. The **plain reduction relation** $\to_{\mathcal{P}}$ is defined as the closure by contexts of the following plain rules:

$$\begin{array}{rcll}
\mathtt{L}\langle\lambda x.t\rangle\, u & \mapsto_{\mathtt{dB}} & \mathtt{L}\langle t[x\setminus u]\rangle & \mathsf{fc}(u,\mathtt{L}) \\
\mathtt{L}\langle\mu\alpha.c\rangle\, u & \mapsto_{\mathtt{dM}} & \mathtt{L}\langle\mu\alpha'.c[\![\alpha\setminus^{\alpha'} u]\!]\rangle & \mathsf{fc}(u,\mathtt{L}),\alpha' \text{ fresh} \\
\mathtt{LCC}\langle[\alpha]\, t\rangle[\![\alpha\setminus^{\alpha'} s]\!] & \mapsto_{\mathtt{N}} & \mathtt{LCC}\langle[\alpha']\, t :: s\rangle & \alpha\notin\mathtt{LCC},\alpha\notin t \\
\mathtt{LCC}\langle c'[\![\beta\setminus^{\alpha} s']\!]\rangle[\![\alpha\setminus^{\alpha'} s]\!] & \mapsto_{\mathtt{C}} & \mathtt{LCC}\langle c'[\![\beta\setminus^{\alpha'} s'\cdot s]\!]\rangle & \alpha\notin\mathtt{LCC},\alpha\notin c',\alpha\notin s'
\end{array}$$

i.e.

$$\to_{\mathcal{P}} \triangleq \mathtt{O}_{\mathtt{t}}\langle\mapsto_{\mathtt{dB}}\cup\mapsto_{\mathtt{dM}}\rangle\cup\mathtt{O}_{\mathtt{c}}\langle\mapsto_{\mathtt{N}}\cup\mapsto_{\mathtt{C}}\rangle$$

The set of **plain forms** of the ΛM-calculus is given by $\mathcal{NF}_{\mathcal{P}}\triangleq\mathcal{NF}_{\mathtt{dB},\mathtt{dM},\mathtt{N},\mathtt{C}}$. Moreover, the relation $\to_{\mathcal{P}}$ is terminating and confluent.

Theorem 5.2. *The relation $\to_{\mathcal{P}}$ is terminating.*

Proof. We prove this result by resorting to a polynomial interpretation. Details in Appendix B. □

Theorem 5.3. *The relation $\to_{\mathcal{P}}$ has the diamond property and hence it is confluent.*

Proof. We prove that the relation $\to_{\mathcal{P}}$ has the diamond property by inspecting all possible cases. Details can be found in Appendix B. □

From now on, we will refer to the (unique) **plain form** of an object o as $\mathcal{P}(o)\triangleq\mathsf{nf}_{\mathcal{P}}(o)$.

Definition 5.4. The **meaningful replacement reduction relation** $\to_{\mathtt{R}^\bullet}$ is defined as the closure by contexts of the non-linear rules $\mathtt{Rnl}$, $\mathtt{Nnl}$, and $\mathtt{Cnl}$, *i.e.*

$$\to_{\mathtt{R}^\bullet}\triangleq\mathtt{O}_{\mathtt{c}}\langle\mapsto_{\mathtt{Rnl}}\cup\mapsto_{\mathtt{Nnl}}\cup\mapsto_{\mathtt{Cnl}}\rangle$$

The **meaningful reduction relation** $\rightsquigarrow$ for the ΛM-calculus on plain forms is given by:

$$o\rightsquigarrow o' \quad\text{iff}\quad o\to_{\mathtt{S},\mathtt{R}^\bullet} p \text{ and } o'=\mathcal{P}(p)$$

We occasionally use $\rightsquigarrow_{\mathtt{S}}$ and $\rightsquigarrow_{\mathtt{R}^\bullet}$ to make explicit which rule is used in a $\rightsquigarrow$-step.

$$(\lambda y.t)[x \setminus u] \underset{y \notin u}{\simeq_{\tilde{\sigma}_1}} \lambda y.t[x \setminus u]$$

$$(t\,v)[x \setminus u] \underset{x \notin v}{\simeq_{\tilde{\sigma}_2}} t[x \setminus u]\,v$$

$$(\mu\beta.[\alpha]\,t)[x \setminus u] \underset{\beta \notin u}{\simeq_{\tilde{\sigma}_3}} \mu\beta.[\alpha]\,t[x \setminus u]$$

$$[\alpha']\,\mu\alpha''.([\beta']\,\mu\beta''.c[\![\beta \setminus^{\beta''} v]\!])[\![\alpha \setminus^{\alpha''} u]\!] \underset{\alpha \notin v, \alpha'' \notin v, \beta \notin u, \beta'' \notin u}{\simeq_{\tilde{\sigma}_4}} [\beta']\,\mu\beta''.([\alpha']\,\mu\alpha''.c[\![\alpha \setminus^{\alpha''} u]\!])[\![\beta \setminus^{\beta''} v]\!]$$

$$[\alpha']\,\mu\alpha''.([\beta']\,\lambda y.\mu\beta.c)[\![\alpha \setminus^{\alpha''} u]\!] \underset{y \notin u, \beta \notin u, \alpha'' \notin u}{\simeq_{\tilde{\sigma}_5}} [\beta']\,\lambda y.\mu\beta.[\alpha']\,\mu\alpha''.c[\![\alpha \setminus^{\alpha''} u]\!]$$

$$[\alpha']\,\lambda x.\mu\alpha.[\beta']\,\lambda y.\mu\beta.c \simeq_{\tilde{\sigma}_6} [\beta']\,\lambda y.\mu\beta.[\alpha']\,\lambda x.\mu\alpha.c$$

$$\mu\alpha.[\alpha]\,t \underset{\alpha \notin t}{\simeq_{\tilde{\sigma}_7}} t$$

$$[\beta]\,\mu\alpha.c \simeq_{\tilde{\sigma}_8} c\{\alpha \setminus \beta\}$$

$$t[y \setminus v][x \setminus u] \underset{y \notin u, x \notin v}{\simeq_{\tilde{\sigma}_9}} t[x \setminus u][y \setminus v]$$

Figure 4: $\tilde{\sigma}$-equivalence on ΛM-objects.

For example, $([\alpha]\,\lambda x.\mu\gamma.[\alpha]\,\lambda y.\mu\delta.c)[\![\alpha \setminus^{\alpha'} u]\!] \rightarrow_{\mathtt{R}\bullet} [\alpha']\,(\lambda x.\mu\gamma.[\alpha']\,(\lambda y.\mu\delta.c')\,u)\,u \twoheadrightarrow_{\mathcal{P}} [\alpha']\,(\mu\gamma.[\alpha']\,(\mu\delta.c'')[y \setminus u])[x \setminus u]$.

In the classical case, a first attempt to obtaining a strong bisimulation is to consider the rules that result from taking the $\mathcal{P}$-normal form at each side of those from Laurent's σ-equivalence relation (*cf.* Figure 3). The resulting relation $\simeq_{\tilde{\sigma}}$ on ΛM-objects is depicted in Figure 4. This equational theory would be a natural candidate for our strong bisimulation, but unfortunately it is not the case. As discussed in the introduction, the rule $\simeq_{\tilde{\sigma}_8}$ breaks strong bisimulation, so we are thus forced to remove it. But we cannot remove it completely, as it is required for firing linear redexes [Lau03, Proposition 40]. It is also required for swapping explicit renamings in order to close strong bisimulation diagrams (*cf.* Theorem 5.9), this aspect of ρ is incorporated as the rule $\simeq_{\mathtt{exren}}$ in our upcoming σ equivalence.

Linear Contexts. Linear contexts turn out to be very useful when decomposing the rewriting rule $\mathtt{R}$ (*cf.* Section 4.4). Here they are used once again to reduce the amount of necessary rules for the equivalence relation. Note that linear contexts are not only used to support commutation of explicit substitution but also for explicit replacement.

Indeed, the equations $\simeq_{\tilde{\sigma}_1}$, $\simeq_{\tilde{\sigma}_2}$, $\simeq_{\tilde{\sigma}_3}$ and $\simeq_{\tilde{\sigma}_9}$ in Figure 4 are generalised into a single equation reflecting the fact that an explicit substitution commutes with linear contexts. Something similar can be stated for rules $\simeq_{\tilde{\sigma}_4}$ and $\simeq_{\tilde{\sigma}_5}$, between linear contexts and explicit replacement. Moreover, linear contexts can be skipped by any explicit operator (substitution or replacement) as long as they are independent, *i.e.* no undesired capture of free variables/names takes place.

Therefore, we introduce into our equivalence relation $\simeq$ three equations that in turn replace rules $\simeq_{\tilde{\sigma}_1}$, $\simeq_{\tilde{\sigma}_2}$, $\simeq_{\tilde{\sigma}_3}$, $\simeq_{\tilde{\sigma}_4}$, $\simeq_{\tilde{\sigma}_5}$ and $\simeq_{\tilde{\sigma}_9}$ from Figure 4, while extending its behavior to explicit renaming as well:

$$\begin{array}{rcll} \mathtt{LTT}\langle t\rangle[x \setminus u] & \simeq_{\mathtt{exsubs}} & \mathtt{LTT}\langle t[x \setminus u]\rangle & x \notin \mathtt{LTT}, \mathsf{fc}(u, \mathtt{LTT}) \\ \mathtt{LCC}\langle c\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \simeq_{\mathtt{exrepl}} & \mathtt{LCC}\langle c[\![\alpha \setminus^{\alpha'} s]\!]\rangle & \alpha \notin \mathtt{LCC}, \mathsf{fc}(\alpha', \mathtt{LCC}), \mathsf{fc}(s, \mathtt{LCC}) \\ [\beta]\,\mu\alpha.\mathtt{LCC}\langle c\rangle & \simeq_{\mathtt{exren}} & \mathtt{LCC}\langle[\beta]\,\mu\alpha.c\rangle & \alpha \notin \mathtt{LCC}, \mathsf{fc}(\beta, \mathtt{LCC}) \end{array}$$

Structural Equivalence. We are now ready to present $\simeq$.

Definition 5.5 (Structural Equivalence over Plain Forms)**.** The *structural equivalence relation* $\simeq$ is the least congruence relation over plain forms of the ΛM-calculus generated by the rules in Figure 5.

$$\begin{array}{rcll} \mathtt{LTT}\langle t\rangle[x \setminus u] & \simeq_{\mathtt{exsubs}} & \mathtt{LTT}\langle t[x \setminus u]\rangle & x \notin \mathtt{LTT}, \mathsf{fc}(u, \mathtt{LTT}) \\ \mathtt{LCC}\langle c\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \simeq_{\mathtt{exrepl}} & \mathtt{LCC}\langle c[\![\alpha \setminus^{\alpha'} s]\!]\rangle & \alpha \notin \mathtt{LCC}, \mathsf{fc}(\alpha', \mathtt{LCC}), \mathsf{fc}(s, \mathtt{LCC}) \\ [\beta]\,\mu\alpha.\mathtt{LCC}\langle c\rangle & \simeq_{\mathtt{exren}} & \mathtt{LCC}\langle[\beta]\,\mu\alpha.c\rangle & \alpha \notin \mathtt{LCC}, \mathsf{fc}(\beta, \mathtt{LCC}) \\ [\alpha']\,\lambda x.\mu\alpha.[\beta']\,\lambda y.\mu\beta.c & \simeq_{\mathtt{ppop}} & [\beta']\,\lambda y.\mu\beta.[\alpha']\,\lambda x.\mu\alpha.c & \beta \neq \alpha', \alpha \neq \beta' \\ \mu\alpha.[\alpha]\,t & \simeq_{\theta} & t & \alpha \notin t \end{array}$$

Figure 5: Structural equivalence $\simeq$ on ΛM-objects.

First a result on commutation between linear contexts and explicit operators:

Lemma 5.6.
(1) *Let* $t \in \mathbb{T}_{\Lambda M}$*,* $\mathtt{L}$ *be a substitution context and* $\mathtt{LTT}$ *a linear context. Then,* $\mathcal{P}(\mathtt{L}\langle\mathtt{LTT}\langle t\rangle\rangle) \simeq \mathcal{P}(\mathtt{LTT}\langle\mathtt{L}\langle t\rangle\rangle)$ *if* $\mathsf{bv}(\mathtt{L}) \notin \mathtt{LTT}$ *and* $\mathsf{fc}(\mathtt{L}, \mathtt{LTT})$*.*
(2) *Let* $c \in \mathbb{C}_{\Lambda M}$*,* $\mathtt{R}$ *be a repl./ren. context and* $\mathtt{LCC}$ *a linear context. Then,* $\mathcal{P}(\mathtt{R}\langle\mathtt{LCC}\langle c\rangle\rangle) \simeq \mathcal{P}(\mathtt{LCC}\langle\mathtt{R}\langle c\rangle\rangle)$ *if* $\mathsf{bn}(\mathtt{R}) \notin \mathtt{LCC}$ *and* $\mathsf{fc}(\mathtt{R}, \mathtt{LCC})$*.*

Proof. Each case is proved by induction on $\mathtt{L}$ or $\mathtt{R}$ respectively, using some auxiliary results. Details in the Appendix C. □

Note that $\simeq$ is not a congruence on arbitrary terms but on plain forms. For example, $\mu\alpha.[\alpha]\,x$ and x are both in plain form and moreover $\mu\alpha.[\alpha]\,x \simeq x$. However, $(\mu\alpha.[\alpha]\,x)\,y \not\simeq x\,y$ since $(\mu\alpha.[\alpha]\,x)\,y$ is not a plain form. Nevertheless, $\mathcal{P}((\mu\alpha.[\alpha]\,x)\,y) = \mu\alpha.[\alpha]\,x\,y \simeq x\,y = \mathcal{P}(x\,y)$. More generally:

Lemma 5.7. *Let* $o, o' \in \mathbb{O}_{\Lambda M}$*. If* $o \simeq o'$*, then for all context* $\mathtt{O}$ *of appropriate sort,* $\mathcal{P}(\mathtt{O}\langle o\rangle) \simeq \mathcal{P}(\mathtt{O}\langle o'\rangle)$*.*

Proof. By induction on the size of $\mathtt{O}$. Details in the Appendix D. □

Notice that $o \simeq o'$ implies $\mathsf{fv}(o) = \mathsf{fv}(o')$ and $\mathsf{fn}(o) = \mathsf{fn}(o')$. Moreover, all the rules from Figure 4 except $\simeq_{\tilde{\sigma}_8}$ are indeed admissible:

- $\simeq_{\tilde{\sigma}_1}$: $(\lambda y.t)[x \setminus u] \simeq_{\mathtt{exsubs}} \lambda y.t[x \setminus u]$ with $y \notin u$ and $\mathtt{LTT} = \lambda y.\Box$.
- $\simeq_{\tilde{\sigma}_2}$: $(t\,v)[x \setminus u] \simeq_{\mathtt{exsubs}} t[x \setminus u]\,v$ with $x \notin v$ and $\mathtt{LTT} = \Box\,v$.

- $\simeq_{\tilde{\sigma}_3}$: $(\mu\beta.[\alpha]\,t)[x \setminus u] \simeq_{\texttt{exsubs}} \mu\beta.[\alpha]\,t[x \setminus u]$ with $\alpha \notin u$ and $\texttt{LTT} = \mu\beta.[\alpha]\,\Box$.
- $\simeq_{\tilde{\sigma}_4}$: $[\alpha']\,\mu\alpha''.([\beta']\,\mu\beta''.c[\![\beta \setminus^{\beta''} v]\!])[\![\alpha \setminus^{\alpha''} u]\!] \simeq [\beta']\,\mu\beta''.([\alpha']\,\mu\alpha''.c[\![\alpha \setminus^{\alpha''} u]\!])[\![\beta \setminus^{\beta''} v]\!]$ with $\alpha \notin v, \alpha'' \notin v, \beta \notin u, \beta'' \notin u, \beta'' \neq \alpha', \alpha'' \neq \beta'$. This case is particularly interesting since rules $\simeq_{\texttt{exrepl}}$ and $\simeq_{\texttt{exren}}$ play a key role in it:

$$\begin{array}{ll} [\alpha']\,\mu\alpha''.([\beta']\,\mu\beta''.c[\![\beta \setminus^{\beta''} v]\!])[\![\alpha \setminus^{\alpha''} u]\!] & \simeq_{\texttt{exrepl}} \\ [\alpha']\,\mu\alpha''.[\beta']\,\mu\beta''.c[\![\alpha \setminus^{\alpha''} u]\!][\![\beta \setminus^{\beta''} v]\!] & \simeq_{\texttt{exren}} \\ [\beta']\,\mu\beta''.[\alpha']\,\mu\alpha''.c[\![\alpha \setminus^{\alpha''} u]\!][\![\beta \setminus^{\beta''} v]\!] & \simeq_{\texttt{exrepl}} \\ [\beta']\,\mu\beta''.([\alpha']\,\mu\alpha''.c[\![\alpha \setminus^{\alpha''} u]\!])[\![\beta \setminus^{\beta''} v]\!] & \end{array}$$

- $\simeq_{\tilde{\sigma}_5}$: $[\alpha']\,\mu\alpha''.([\beta']\,\lambda y.\mu\beta.c)[\![\alpha \setminus^{\alpha''} u]\!] \simeq [\beta']\,\lambda y.\mu\beta.[\alpha']\,\mu\alpha''.c[\![\alpha \setminus^{\alpha''} u]\!]$ with $y \notin u, \beta \notin u, \beta'' \notin u, \beta'' \neq \alpha', \alpha'' \neq \beta'$. As in the previous case, we can conclude thanks to rules $\simeq_{\texttt{exrepl}}$ and $\simeq_{\texttt{exren}}$:

$$\begin{array}{ll} [\alpha']\,\mu\alpha''.([\beta']\,\lambda y.\mu\beta.c)[\![\alpha \setminus^{\alpha''} u]\!] & \simeq_{\texttt{exrepl}} \\ [\alpha']\,\mu\alpha''.[\beta']\,\lambda y.\mu\beta.c[\![\alpha \setminus^{\alpha''} u]\!] & \simeq_{\texttt{exren}} \\ [\beta']\,\lambda y.\mu\beta.[\alpha']\,\mu\alpha''.c[\![\alpha \setminus^{\alpha''} u]\!] & \end{array}$$

- $\simeq_{\tilde{\sigma}_6}$: $[\alpha']\,\lambda x.\mu\alpha.[\beta']\,\lambda y.\mu\beta.c \simeq_{\texttt{ppop}} [\beta']\,\lambda y.\mu\beta.[\alpha']\,\lambda x.\mu\alpha.c$.
- $\simeq_{\tilde{\sigma}_7}$: $\mu\alpha.[\alpha]\,t \simeq_{\theta} t$.
- $\simeq_{\tilde{\sigma}_9}$: $t[y \setminus v][x \setminus u] \simeq_{\texttt{exsubs}} t[x \setminus u][y \setminus v]$ with $y \notin u$, $x \notin v$ and $\texttt{LTT} = \Box[y \setminus v]$.

Hence, commutation rules $\simeq_{\tilde{\sigma}_1}$, $\simeq_{\tilde{\sigma}_2}$, $\simeq_{\tilde{\sigma}_3}$ and $\simeq_{\tilde{\sigma}_9}$ from Figure 4 are replaced by $\simeq_{\texttt{exsubs}}$, while $\simeq_{\texttt{exrepl}}$ and $\simeq_{\texttt{exren}}$ replace both $\simeq_{\tilde{\sigma}_4}$ and $\simeq_{\tilde{\sigma}_5}$, and $\simeq_{\tilde{\sigma}_8}$ is discarded. Only rules $\simeq_{\tilde{\sigma}_6}$ and $\simeq_{\tilde{\sigma}_7}$ remain unaltered, here called $\simeq_{\texttt{ppop}}$ (for pop/pop) and $\simeq_{\theta}$ respectively (see [Lau03] for the origin of these names). The following table summarises the correspondence between the different rules:

Captured rule from $\tilde{\sigma}$-equivalence	**New rule**
$\simeq_{\tilde{\sigma}_1}$, $\simeq_{\tilde{\sigma}_2}$, $\simeq_{\tilde{\sigma}_3}$ and $\simeq_{\tilde{\sigma}_9}$	$\simeq_{\texttt{exsubs}}$
$\simeq_{\tilde{\sigma}_4}$ and $\simeq_{\tilde{\sigma}_5}$	$\simeq_{\texttt{exrepl}}$ and $\simeq_{\texttt{exren}}$
$\simeq_{\tilde{\sigma}_6}$	$\simeq_{\texttt{ppop}}$
$\simeq_{\tilde{\sigma}_8}$	
$\simeq_{\tilde{\sigma}_7}$	$\simeq_{\theta}$

Strong Bisimulation Result. The resulting equivalence relation $\simeq$ is in fact a strong bisimulation with respect to the notion of meaningful computation, as we will show. First a simple result relating plain reduction and structural equivalence.

Lemma 5.8. *Let $u, s, o \in \mathbb{O}_{\Lambda M}$ in $\mathcal{P}$-normal form such that $o \simeq o'$, $u \simeq u'$ and $s \simeq s'$, where u, u' are terms and s, s' are stacks. Then,*

(1) $\mathcal{P}(o\{x \setminus u\}) \simeq \mathcal{P}(o'\{x \setminus u\})$ *and* $\mathcal{P}(o\{x \setminus u\}) \simeq \mathcal{P}(o\{x \setminus u'\})$.
(2) $\mathcal{P}(o\{\!\{\alpha \setminus^{\alpha'} s\}\!\}) \simeq \mathcal{P}(o'\{\!\{\alpha \setminus^{\alpha'} s\}\!\})$ *and* $\mathcal{P}(o\{\!\{\alpha \setminus^{\alpha'} s\}\!\}) \simeq \mathcal{P}(o\{\!\{\alpha \setminus^{\alpha'} s'\}\!\})$.

Proof. Each case is proved by induction on $o \simeq o'$. Details in the Appendix D. □

We are now able to state the promised result, namely, the fact that $\simeq$ is a strong bisimulation with respect to the meaningful computation relation.

Theorem 5.9. *Let $o, p \in \mathbb{O}_{\Lambda M}$. If $o \simeq p$ and $o \rightsquigarrow o'$, then there exists p' such that $p \rightsquigarrow p'$ and $o' \simeq p'$.*

Proof. The proof is by induction on $o \simeq p$ and uses Lemma 5.6, Lemma 5.7 and Lemma 5.8. All the details are in the Appendix D. □

Example 5.10. We illustrate the previous theorem with the following example. Let $o = o_0[\![\alpha \setminus^{\alpha'} u]\!] \simeq_{\mathtt{ppop}} p_0[\![\alpha \setminus^{\alpha'} u]\!] = p$, where $o_0 = [\alpha]\,\lambda x.\mu\gamma.[\alpha]\,\lambda y.\mu\delta.c \simeq_{\mathtt{ppop}} [\alpha]\,\lambda y.\mu\delta.[\alpha]\,\lambda x.\mu\gamma.c = p_0$. Notice in particular that o_0, o, p_0 and p are all plain forms. We have

$$\begin{array}{ccc}
o = ([\alpha]\,\lambda x.\mu\gamma.[\alpha]\,\lambda y.\mu\delta.c)[\![\alpha \setminus^{\alpha'} u]\!] & \simeq_{\mathtt{ppop}} & ([\alpha]\,\lambda y.\mu\delta.[\alpha]\,\lambda x.\mu\gamma.c)[\![\alpha \setminus^{\alpha'} u]\!] = p \\
\downarrow_{\mathtt{R}^\bullet} & & \downarrow_{\mathtt{R}^\bullet} \\
[\alpha']\,(\lambda x.\mu\gamma.[\alpha']\,(\lambda y.\mu\delta.c')\,u)\,u & & [\alpha']\,(\lambda y.\mu\delta.[\alpha']\,(\lambda x.\mu\gamma.c')\,u)\,u \\
\twoheaddownarrow_{\mathcal{P}} & & \twoheaddownarrow_{\mathcal{P}} \\
o' = [\alpha']\,(\mu\gamma.[\alpha']\,(\mu\delta.c'')[y \setminus u])[x \setminus u] & \simeq_{\mathtt{exsubs,exren}} & [\alpha']\,(\mu\delta.[\alpha']\,(\mu\gamma.c'')[x \setminus u])[y \setminus u] = p'
\end{array}$$

so that $o \rightsquigarrow o'$ and $p \rightsquigarrow p'$. We conclude $o' \simeq p'$ as follows:

$$\begin{array}{ll}
[\alpha']\,(\mu\gamma.[\alpha']\,(\mu\delta.c'')[y \setminus u])[x \setminus u] & \simeq_{\mathtt{exsubs}} \\
[\alpha']\,\mu\gamma.[\alpha']\,(\mu\delta.c'')[y \setminus u][x \setminus u] & \simeq_{\mathtt{exren}} \\
[\alpha']\,(\mu\delta.[\alpha']\,\mu\gamma.c'')[y \setminus u][x \setminus u] & \simeq_{\mathtt{exsubs}} \\
[\alpha']\,(\mu\delta.[\alpha']\,(\mu\gamma.c'')[x \setminus u])[y \setminus u] &
\end{array}$$

Note the use of $\simeq_{\mathtt{exren}}$ to swap the occurrences of $[\alpha']\,\mu\gamma.\ldots$ and $[\alpha']\,\mu\delta.\ldots$.

6. Correspondence result

In this section we relate our bisimulation $\simeq$ on ΛM-objects to Laurent's original σ-equivalence on $\lambda\mu$-objects. More precisely, we show that $\simeq$ can be seen as Laurent's σ-equivalence devoid of axiom $\simeq_{\sigma_8}$ (usually called ρ-axiom) but enriched with three additional axioms $\simeq_{\tau_8}$, $\simeq_{\tau_9}$, and $\simeq_{\tau_{10}}$ (also called (ren/ren), (ren/pop) and (ren/push) resp.), which are admissible in Laurent's σ-equivalence. Indeed, all of $\simeq_{\tau_8}$, $\simeq_{\tau_9}$, and $\simeq_{\tau_{10}}$ are derivable from $\simeq_{\sigma_8}$, but the converse does not hold. In this sense, $\simeq_\tau$ can be seen as a restriction of Laurent's σ-equivalence.

Definition 6.1. The new $\simeq_\tau$-equivalence for $\lambda\mu$-objects is depicted in Figure 6.

The new $\simeq_\tau$-equivalence is built by removing axiom $\simeq_{\sigma_8}$ from Laurent's σ-equivalence, and by adding $\simeq_{\tau_8}$, $\simeq_{\tau_9}$, and $\simeq_{\tau_{10}}$. Notice that axiom $\simeq_{\tau_i}$ in Figure 6 is exactly the same as $\simeq_{\sigma_i}$ in Figure 3 for $i = 1 \ldots 7$. Moreover, notice that none of $\simeq_{\tau_8}$, $\simeq_{\tau_9}$, and $\simeq_{\tau_{10}}$ erase or introduce explicit renamings.

Some axioms of the new relation $\simeq_\tau$ can be generalized to several arguments. For that, we use the meta-notation $u :: s$ introduced in Section 4, in this case denoting a term in $\mathbb{T}_{\lambda\mu}$, resulting from the application of u to a stack s of terms in $\mathbb{T}_{\lambda\mu}$, *i.e.* if $s = t_0 \cdot \ldots \cdot t_n$ and $t_0 \ldots t_n \in \mathbb{T}_{\lambda\mu}$, then $u :: s \triangleq u\,t_0 \ldots t_n$ denotes a term in $\mathbb{T}_{\lambda\mu}$.

Lemma 6.2. *Let $t, u \in \mathbb{T}_{\lambda\mu}$ and $c \in \mathbb{C}_{\lambda\mu}$. Let s be a stack of terms in $\mathbb{T}_{\lambda\mu}$. Let $x \notin s, \alpha \notin s', \beta \notin s, \beta \neq \alpha', \alpha \neq \beta'$. Then,*

(1) $((\lambda x.t)\,u) :: s \simeq_\tau (\lambda x.t :: s)\,u$

(2) $[\alpha']\,(\mu\alpha.[\beta']\,(\mu\beta.c) :: s') :: s \simeq_\tau [\beta']\,(\mu\beta.[\alpha']\,(\mu\alpha.c) :: s) :: s'$

$$\begin{array}{rcll}
(\lambda y.\lambda x.t)\, v & \simeq_{\tau_1} & \lambda x.(\lambda y.t)\, v & x \notin v \\
(\lambda x.t\, v)\, u & \simeq_{\tau_2} & (\lambda x.t)\, u\, v & x \notin v \\
(\lambda x.\mu\alpha.[\beta]\, t)\, u & \simeq_{\tau_3} & \mu\alpha.[\beta]\, (\lambda x.t)\, u & \alpha \notin u \\
[\alpha']\, (\mu\alpha.[\beta']\, (\mu\beta.c)\, v)\, u & \simeq_{\tau_4} & [\beta']\, (\mu\beta.[\alpha']\, (\mu\alpha.c)\, u)\, v & \alpha \notin v, \beta \notin u, \beta \neq \alpha', \alpha \neq \beta' \\
[\alpha']\, (\mu\alpha.[\beta']\, \lambda y.\mu\beta.c)\, u & \simeq_{\tau_5} & [\beta']\, \lambda y.\mu\beta.[\alpha']\, (\mu\alpha.c)\, u & y \notin u, \beta \notin u, \beta \neq \alpha', \alpha \neq \beta' \\
[\alpha']\, \lambda x.\mu\alpha.[\beta']\, \lambda y.\mu\beta.c & \simeq_{\tau_6} & [\beta']\, \lambda y.\mu\beta.[\alpha']\, \lambda x.\mu\alpha.c & \beta \neq \alpha', \alpha \neq \beta' \\
\mu\alpha.[\alpha]\, t & \simeq_{\tau_7} & t & \alpha \notin t \\
[\alpha']\, \mu\alpha.[\beta']\, \mu\beta.c & \simeq_{\tau_8} & [\beta']\, \mu\beta.[\alpha']\, \mu\alpha.c & \beta \neq \alpha', \alpha \neq \beta' \\
[\alpha']\, \mu\alpha.[\beta']\, \lambda y.\mu\beta.c & \simeq_{\tau_9} & [\beta']\, \lambda y.\mu\beta.[\alpha']\, \mu\alpha.c & \beta \neq \alpha', \alpha \neq \beta' \\
[\alpha']\, \mu\alpha.[\beta']\, (\mu\beta.c)\, v & \simeq_{\tau_{10}} & [\beta']\, (\mu\beta.[\alpha']\, \mu\alpha.c)\, v & \alpha \notin v, \beta \neq \alpha', \alpha \neq \beta'
\end{array}$$

Figure 6: New $\simeq_\tau$-equivalence for $\lambda\mu$-objects

(3) $[\alpha']\, (\mu\alpha.[\beta']\, \lambda x.\mu\beta.c) :: s \simeq_\tau [\beta']\, \lambda x.\mu\beta.[\alpha']\, (\mu\alpha.c) :: s$
(4) $[\alpha']\, (\mu\alpha.[\beta']\, \mu\beta.c) :: s \simeq_\tau [\beta']\, \mu\beta.[\alpha']\, (\mu\alpha.c) :: s$

Proof. The proof is in Appendix E. □

To obtain the desired correspondence between $\simeq$ on ΛM-objects and $\simeq_\tau$ on $\lambda\mu$-objects, it is necessary to relate the sets $\mathbb{O}_{\Lambda M}$ and $\mathbb{O}_{\lambda\mu}$. We do so by means of an **expansion function** $\mathtt{e}(\_)$, that eliminates the explicit operators of an object by dBdM-expansions:

$$\begin{array}{rclrcl}
\mathtt{e}(x) & \triangleq & x & \mathtt{e}(t[x \setminus u]) & \triangleq & (\lambda x.\mathtt{e}(t))\, \mathtt{e}(u) \\
\mathtt{e}(t\, u) & \triangleq & \mathtt{e}(t)\, \mathtt{e}(u) & \mathtt{e}([\alpha]\, t) & \triangleq & [\alpha]\, \mathtt{e}(t) \\
\mathtt{e}(\lambda x.t) & \triangleq & \lambda x.\mathtt{e}(t) & \mathtt{e}(c[\![\alpha \setminus^{\alpha'} s]\!]) & \triangleq & [\alpha']\, (\mu\alpha.\mathtt{e}(c)) :: \mathtt{e}(s) \\
\mathtt{e}(\mu\alpha.c) & \triangleq & \mu\alpha.\mathtt{e}(c) & \mathtt{e}(t \cdot s) & \triangleq & \mathtt{e}(t) \cdot \mathtt{e}(s)
\end{array}$$

Note that $\mathtt{e}(\_)$ is not the left-inverse of the plain form $\mathcal{P}(\_)$, *i.e.* given $o \in \mathbb{O}_{\lambda\mu}$, it is not necessarily the case $\mathtt{e}(\mathcal{P}(o)) = o$. For example, take $o = (\mu\alpha.[\alpha]\, x\, (\mu\beta.[\alpha]\, w))\, y\, z$. Then, $\mathcal{P}(o) = \mu\alpha'.([\alpha]\, x\, (\mu\beta.[\alpha]\, w))[\![\alpha \setminus^{\alpha'} y \cdot z]\!]$ while $\mathtt{e}(\mathcal{P}(o)) = \mu\alpha'.[\alpha']\, (\mu\alpha.[\alpha]\, x\, (\mu\beta.[\alpha]\, w))\, y\, z$. However, it yields an equivalent object thanks to rule $\simeq_{\tau_7}$.

Some basic properties of the expansion function are stated below.

Lemma 6.3. *Let $t, u \in \mathbb{T}_{\lambda\mu}$ and $o \in \mathbb{O}_{\Lambda M}$. Let* $\mathtt{L}$ *be a substitution context. Then,*
(1) $\mathtt{e}(\mathtt{L}\langle t[x \setminus u]\rangle) \simeq_\tau \mathtt{e}(\mathtt{L}\langle t\rangle[x \setminus u])$
(2) $\mathtt{e}(\mathtt{L}\langle t\rangle\, u) \simeq_\tau \mathtt{e}(\mathtt{L}\langle t\, u\rangle)$
(3) $\mathtt{e}(\mathtt{e}(o)) = \mathtt{e}(o)$
(4) $\mathtt{e}(\mathtt{L}\langle t\rangle) = \mathtt{e}(\mathtt{L}\langle \mathtt{e}(t)\rangle)$
(5) *If $t, t' \in \mathbb{T}_{\lambda\mu}$, then $t \simeq_\tau t'$ implies* $\mathtt{e}(\mathtt{L}\langle t\rangle) \simeq_\tau \mathtt{e}(\mathtt{L}\langle t'\rangle)$

Proof. The first and second point are by induction on $\mathtt{L}$. The third point is by straightforward induction on o. The fourth point is by induction on $\mathtt{L}$ using the third point. The last one is by induction on $\mathtt{L}$ using the fact that $\simeq_\tau$ is a congruence. □

The expansion function allows to $\simeq_\tau$-equate ΛM-objects that are related by the reduction $\to_{\mathcal{P}}$ (*cf.* Definition 5.1). We start with the more subtle cases $\mapsto_{\mathtt{N}}$ and $\mapsto_{\mathtt{C}}$.

Lemma 6.4. *Let $t \in \mathbb{T}_{\lambda\mu}$ and $c \in \mathbb{C}_{\lambda\mu}$. Let s, s' be stacks and* LCC *be a* ***CC*** *linear context. Then,*
(1) $\mathtt{e}(\mathtt{LCC}\langle [\alpha]\, t\rangle[\![\alpha \setminus^{\alpha'} s]\!]) \simeq_\tau \mathtt{e}(\mathtt{LCC}\langle [\alpha']\, t :: s\rangle)$, *where $\alpha \notin$* LCC *and $\alpha \notin t$.*

(2) $\mathtt{e}(\mathtt{LCC}\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!]) \simeq_{\mathcal{T}} \mathtt{e}(\mathtt{LCC}\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle) = o'$, *where* $\alpha \notin \mathtt{LCC}$, $\alpha \notin c$ *and* $\alpha \notin s'$.

Proof. The proof is in Appendix E. □

Lemma 6.5. *Let* $o \in \mathbb{O}_{\Lambda M}$. *Then* $o \to_{\mathcal{P}} o'$ *implies* $\mathtt{e}(o) \simeq_{\mathcal{T}} \mathtt{e}(o')$

Proof. We only show the base cases:

- $o = \mathtt{L}\langle \lambda x.t\rangle\, u \mapsto_{\mathtt{dB}} \mathtt{L}\langle t[x \setminus u]\rangle = o'$, where $\mathtt{fc}(u, \mathtt{L})$. Then by Lemma 6.3 we have

$$\begin{array}{rclcl} \mathtt{e}(\mathtt{L}\langle \lambda x.t\rangle\, u) & \simeq_{\mathcal{T}} & \mathtt{e}(\mathtt{L}\langle (\lambda x.t)\, u\rangle) & \simeq_{\mathcal{T}} & \mathtt{e}(\mathtt{L}\langle \mathtt{e}((\lambda x.t)\, u)\rangle) \\ & = & \mathtt{e}(\mathtt{L}\langle (\lambda x.\mathtt{e}(t))\, \mathtt{e}(u)\rangle) & = & \mathtt{e}(\mathtt{L}\langle \mathtt{e}(t[x \setminus u])\rangle) \\ & \simeq_{\mathcal{T}} & \mathtt{e}(\mathtt{L}\langle t[x \setminus u]\rangle) & & \end{array}$$

- $o = \mathtt{L}\langle \mu\alpha.c\rangle\, u \mapsto_{\mathtt{dM}} \mathtt{L}\langle \mu\alpha'.c[\![\alpha \setminus^{\alpha'} u]\!]\rangle = o'$, where $\mathtt{fc}(u, \mathtt{L})$ and α' is fresh. Then by Lemma 6.3 we have

$$\begin{array}{rclcl} \mathtt{e}(o) & = & \mathtt{e}(\mathtt{L}\langle \mu\alpha.c\rangle\, u) & \simeq_{\mathcal{T}} & \mathtt{e}(\mathtt{L}\langle (\mu\alpha.c)\, u\rangle) \\ & \simeq_{\mathcal{T}} & \mathtt{e}(\mathtt{L}\langle \mathtt{e}((\mu\alpha.c)\, u)\rangle) & = & \mathtt{e}(\mathtt{L}\langle (\mu\alpha.\mathtt{e}(c))\, \mathtt{e}(u)\rangle) \\ & \simeq_{\mathcal{T}} & \mathtt{e}(\mathtt{L}\langle \mu\alpha'.[\alpha']\, (\mu\alpha.\mathtt{e}(c))\, \mathtt{e}(u)\rangle) & = & \mathtt{e}(\mathtt{L}\langle \mathtt{e}(\mu\alpha'.c[\![\alpha \setminus^{\alpha'} u]\!])\rangle) \\ & = & \mathtt{e}(o') & & \end{array}$$

- $o = \mathtt{LCC}\langle [\alpha]\, t\rangle[\![\alpha \setminus^{\alpha'} s]\!] \mapsto_{\mathtt{N}} \mathtt{LCC}\langle [\alpha']\, t :: s\rangle = o'$, where $\alpha \notin \mathtt{LCC}$ and $\alpha \notin t$. The result follows from Lemma 6.4(1).
- $o = \mathtt{LCC}\langle c'[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!] \mapsto_{\mathtt{C}} \mathtt{LCC}\langle c'[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle = o'$, where $\alpha \notin \mathtt{LCC}$, $\alpha \notin c'$ and $\alpha \notin s'$. The result follows from Lemma 6.4(2). □

We can then conclude that plain forms do not change by expansion.

Corollary 6.6. *Let* $o \in \mathbb{O}_{\Lambda M}$. *Then* $\mathtt{e}(\mathcal{P}(o)) \simeq_{\mathcal{T}} \mathtt{e}(o)$.

We now show that $\simeq_{\mathcal{T}}$-equivalent $\lambda\mu$-objects project into $\simeq$ by means of the plain form.

Lemma 6.7. *Let* $o, p \in \mathbb{O}_{\lambda\mu}$. *Then,* $o \simeq_{\mathcal{T}} p$ *implies* $\mathcal{P}(o) \simeq \mathcal{P}(p)$.

Proof. The proof is in Appendix E. □

For the converse we use the expansion function, *i.e.* $\simeq$-equivalent ΛM-objects project into $\simeq_{\mathcal{T}}$ by means of the expansion function.

Lemma 6.8. *Let* $o, p \in \mathbb{O}_{\Lambda M}$. *Then,* $o \simeq p$ *implies* $\mathtt{e}(o) \simeq_{\mathcal{T}} \mathtt{e}(p)$.

Proof. The proof is in Appendix E. □

The properties above allow to us conclude with the following result.

Theorem 6.9. *Let* $o, p \in \mathbb{O}_{\lambda\mu}$. *Then* $o \simeq_{\mathcal{T}} p$ *iff* $\mathcal{P}(o) \simeq \mathcal{P}(p)$.

Proof.

- $\Rightarrow$) By Lemma 6.7.
- $\Leftarrow$) $\mathcal{P}(o) \simeq \mathcal{P}(p)$ implies $\mathtt{e}(\mathcal{P}(o)) \simeq_{\mathcal{T}} \mathtt{e}(\mathcal{P}(p))$ by Lemma 6.8. From $\mathtt{e}(\mathcal{P}(o)) \simeq_{\mathcal{T}} \mathtt{e}(\mathcal{P}(p))$ we obtain $\mathtt{e}(o) \simeq_{\mathcal{T}} \mathtt{e}(p)$ by Corollary 6.6. Since o, p are pure $\lambda\mu$-terms, then $\mathtt{e}(o) = o$ and $\mathtt{e}(p) = p$. Thus, $o \simeq_{\mathcal{T}} p$. □

Even if this last theorem relates the new $\simeq_\tau$-equivalence to the strong bisimulation $\simeq$ presented in Section 5, the resulting property also explains the relationship between Laurent's σ-equivalence and $\simeq$. Indeed, starting from the fact that ρ-equivalence breaks strong bisimulation (*cf.* example in the introduction), ρ-equivalence is restricted (through our adoption of $\simeq_{\tau_8}$, $\simeq_{\tau_9}$, and $\simeq_{\tau_{10}}$ in lieu of ρ) to its non-erasing and non-duplicating role in swapping names and μ-binders in the new relation $\simeq_\tau$. In this way, we keep the strictly necessary renaming operation of Laurent's original σ-equivalence which is able to materialize a correspondence with our strong bisimulation.

7. Conclusion

This paper is about σ-equivalence in classical logic and the negligible effect it has on the operational behavior of the terms it relates. It refines the $\lambda\mu$-calculus with explicit operators for substitution and replacement, by splitting in particular each of the rules β and μ of $\lambda\mu$ into multiplicative and exponential fragments, thus resulting in the introduction of a new calculus called ΛM. This new presentation of $\lambda\mu$ allows to reformulate σ-equivalence on $\lambda\mu$-terms as a strong bisimulation relation $\simeq$ on ΛM-terms. The main obstacle to extract a bisimulation on ΛM from the original Laurent's σ-equivalence on $\lambda\mu$-terms is axiom $\simeq_\rho$, which leads to σ-equivalence failing to be a strong bisimulation. We learn that we cannot remove $\simeq_\rho$ entirely, since it is needed to close several commutation diagrams in the proof of strong bisimulation. However, a restriction of $\simeq_\rho$ turns out to suffice.

In [KV19], the $\lambda\mu$-calculus is refined to a calculus $\lambda\mu\mathtt{s}$ with explicit operators, together with a *linear* substitution/replacement operational semantics *at a distance*. In contrast to ΛM, $\lambda\mu\mathtt{s}$ does not support composition of explicit replacements. In particular, explicit replacements in $\lambda\mu\mathtt{s}$ are defined on terms, and not on stacks, thus the calculus is not able to capture an appropriate notion of bisimulation such as the one presented in this paper.

Other classical term calculi exist, *e.g.* [CH00, Aud94, Pol04, vBV14], but none of these formalisms decomposes term reduction by means of a fine distinction between multiplicative and exponential rules. Thus, the main ingredients needed to build a strong bisimulation are simply not available. Of particular interest would be obtain a strong bisimulation in the setting of $\lambda\mu\widetilde{\mu}$ [CH00], a calculus inspired from sequent calculus which is constructed as a perfectly symmetric formalism to deal uniformly with CBN and CBV.

Explicit Substitutions are a means of modeling sharing in lambda calculi and hence well suited for capturing call-by-need [ABM14]. We believe ΛM may prove useful in devising a notion of call-by-need for classical computation. Our notation for explicit replacement and the notion of single replacement it supports (*cf.* Section 4.2), would play a crucial role in formulating such a calculus.

A further related reference is [HL10], where PPNs are used to interpret processes from the π-calculus. A precise correspondence is established between PPN and a typed version of the asynchronous π-calculus. Moreover, they show that Laurent's $\simeq_\sigma$ corresponds exactly to structural equivalence of π-calculus processes (Proposition 1 in op.cit). In [LR03] Laurent and Regnier show that there is a precise correspondence between CPS translations from classical calculi (such as $\lambda\mu$) into intuitionistic ones on the one hand, and translations between LLP and LL on the other.

It would be interesting to analyse other rewriting properties of our term language such as preservation of $\lambda\mu$-strong normalisation of the reduction relations $\to_{\Lambda M}$ and $\leadsto$ or confluence of $\leadsto$.

Moreover, following the computational interpretation of *deep inference* provided by the intuitionistic atomic lambda-calculus [GHP13], it would be interesting to investigate a classical extension and its corresponding notion of strong bisimulation. It is also natural to wonder what would be an ideal syntax for classical logic, that is able to capture strong bisimulation by reducing the syntactical axioms to a small and simple set of equations.

Finally, our notion of $\simeq$-equivalence could facilitate proofs of correctness between abstract machines and $\lambda\mu$ (like [ABM14] for lambda-calculus) and help establish whether abstract machines for $\lambda\mu$ are "reasonable" [ABM14].

Acknowledgment

To the reviewers for detailed feedback. In particular, for suggesting an alternative proof of Theorem 5.2 and a simplified syntax for ΛM regarding explicit renamings.

References

[ABKL14] Beniamino Accattoli, Eduardo Bonelli, Delia Kesner, and Carlos Lombardi. A nonstandard standardization theorem. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 659–670, 2014. doi:10.1145/2535838.2535886.

[ABM14] Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. Distilling abstract machines. In *Proceedings of the 19th ACM SIGPLAN international conference on Functional programming, Gothenburg, Sweden, September 1-3, 2014*, pages 363–376, 2014. doi:10.1145/2628136.2628154.

[Acc18] Beniamino Accattoli. Proof nets and the linear substitution calculus. In *Theoretical Aspects of Computing - ICTAC 2018 - 15th International Colloquium, Stellenbosch, South Africa, October 16-19, 2018, Proceedings*, pages 37–61, 2018. doi:10.1007/978-3-030-02508-3_3.

[AK10] Beniamino Accattoli and Delia Kesner. The structural *lambda*-calculus. In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 381–395. Springer, 2010. doi:10.1007/978-3-642-15205-4_30.

[AK12a] Beniamino Accattoli and Delia Kesner. The permutative λ-calculus. In *Logic for Programming, Artificial Intelligence, and Reasoning - 18th International Conference, LPAR-18, Mérida, Venezuela, March 11-15, 2012. Proceedings*, pages 23–36, 2012. doi:10.1007/978-3-642-28717-6_5.

[AK12b] Beniamino Accattoli and Delia Kesner. Preservation of strong normalisation modulo permutations for the structural lambda-calculus. *Log. Methods Comput. Sci.*, 8(1), 2012. doi:10.2168/LMCS-8(1:28)2012.

[And03] Yuuki Andou. Church-rosser property of a simple reduction for full first-order classical natural deduction. *Ann. Pure Appl. Log.*, 119(1-3):225–237, 2003. doi:10.1016/S0168-0072(02)00051-9.

[Aud94] Philippe Audebaud. Explicit substitutions for the lambda-mu-calculus. Technical Report RR-1994-26, LIP, Ecole Normale Supérieure de Lyon, Lyon, 1994.

[BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, 1998. doi:10.1017/CBO9781139172752.

[CH00] Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000.*, pages 233–243, 2000. doi:10.1145/351240.351262.

[CHL96] Pierre-Louis Curien, Thérèse Hardin, and Jean-Jacques Lévy. Confluence properties of weak and strong calculi of explicit substitutions. *J. ACM*, 43(2):362–397, 1996. doi:10.1145/226643.226675.

[dG94] Philippe de Groote. On the relation between the lambda-mu-calculus and the syntactic theory of sequential control. In *Logic Programming and Automated Reasoning, 5th International Conference, LPAR'94, Kiev, Ukraine, July 16-22, 1994, Proceedings*, pages 31–43, 1994. doi:10.1007/3-540-58216-9_27.

[GHP13] Tom Gundersen, Willem Heijltjes, and Michel Parigot. Atomic lambda calculus: A typed lambda-calculus with explicit sharing. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 311–320, 2013. doi:10.1109/LICS.2013.37.

[Gir87] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987. doi:10.1016/0304-3975(87)90045-4.

[Gir91] Jean-Yves Girard. A new constructive logic: Classical logic. *Math. Struct. Comput. Sci.*, 1(3):255–296, 1991. doi:10.1017/S0960129500001328.

[Gri90] Timothy Griffin. A formulae-as-types notion of control. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages, San Francisco, California, USA, January 1990*, pages 47–58, 1990. doi:10.1145/96709.96714.

[Her94] Hugo Herbelin. A lambda-calculus structure isomorphic to gentzen-style sequent calculus structure. In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers*, volume 933 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 1994. doi:10.1007/BFb0022247.

[HL10] Kohei Honda and Olivier Laurent. An exact correspondence between a typed pi-calculus and polarised proof-nets. *Theor. Comput. Sci.*, 411(22-24):2223–2238, 2010. doi:10.1016/j.tcs.2010.01.028.

[KBV20] Delia Kesner, Eduardo Bonelli, and Andrés Viso. Strong bisimulation for control operators (invited talk). In *28th EACSL Annual Conference on Computer Science Logic, CSL 2020, January 13-16, 2020, Barcelona, Spain*, pages 4:1–4:23, 2020. doi:10.4230/LIPIcs.CSL.2020.4.

[Kes09] Delia Kesner. A theory of explicit substitutions with safe and full composition. *Log. Methods Comput. Sci.*, 5(3), 2009. URL: http://arxiv.org/abs/0905.2539.

[Kes22] Delia Kesner. A fine-grained computational interpretation of girard's intuitionistic proof-nets. *Proc. ACM Program. Lang.*, 6(POPL):1–28, 2022. doi:10.1145/3498669.

[KV19] Delia Kesner and Daniel Ventura. A resource aware semantics for a focused intuitionistic calculus. *Mathematical Structures in Computer Science*, 29(1):93–126, 2019. doi:10.1017/S0960129517000111.

[Lau02] Olivier Laurent. *A study of polarization in logic*. PhD thesis, Université de la Méditerranée - Aix-Marseille II, 2002.

[Lau03] Olivier Laurent. Polarized proof-nets and lambda-μ-calculus. *Theor. Comput. Sci.*, 290(1):161–188, 2003. doi:10.1016/S0304-3975(01)00297-3.

[LR03] Olivier Laurent and Laurent Regnier. About translations of classical logic into polarized linear logic. In *18th IEEE Symposium on Logic in Computer Science (LICS 2003), 22-25 June 2003, Ottawa, Canada, Proceedings*, pages 11–20, 2003. doi:10.1109/LICS.2003.1210040.

[Par92] 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. doi:10.1007/BFb0013061.

[Par93] Michel Parigot. Classical proofs as programs. In *Computational Logic and Proof Theory, Third Kurt Gödel Colloquium, KGC'93, Brno, Czech Republic, August 24-27, 1993, Proceedings*, pages 263–276, 1993. doi:10.1007/BFb0022575.

[Pol04] Emmanuel Polonovski. *Subsitutions explicites, logique et normalisation. (Explicit substitutions, logic and normalization)*. PhD thesis, Paris Diderot University, France, 2004.

[Reg94] Laurent Regnier. Une équivalence sur les lambda-termes. *Theor. Comput. Sci.*, 126(2):281–292, 1994. doi:10.1016/0304-3975(94)90012-4.

[vBV14] Steffen van Bakel and Maria Grazia Vigliotti. A fully-abstract semantics of lambda-mu in the pi-calculus. In *Proceedings Fifth International Workshop on Classical Logic and Computation, CL&C 2014, Vienna, Austria, July 13, 2014*, pages 33–47, 2014. doi:10.4204/EPTCS.164.3.

Appendix A. Confluence of the ΛM-calculus

To prove confluence of the ΛM-calculus we use the interpretation method [CHL96], where ΛM is projected into the $\lambda\mu$-calculus.

Definition A.1. The **projection** $\_^{\downarrow}$ from ΛM-objects to $\lambda\mu$-objects is defined as

$$\begin{array}{rcl} x^{\downarrow} & \triangleq & x \\ (t\,u)^{\downarrow} & \triangleq & t^{\downarrow}\,u^{\downarrow} \\ (\lambda x.t)^{\downarrow} & \triangleq & \lambda x.t^{\downarrow} \\ (\mu\alpha.c)^{\downarrow} & \triangleq & \mu\alpha.c^{\downarrow} \\ (t[x \setminus u])^{\downarrow} & \triangleq & t^{\downarrow}\{x \setminus u^{\downarrow}\} \end{array} \qquad \begin{array}{rcl} ([\alpha]\,t)^{\downarrow} & \triangleq & [\alpha]\,t^{\downarrow} \\ (c[\![\alpha \setminus^{\alpha'} s]\!])^{\downarrow} & \triangleq & c^{\downarrow}\{\!\{\alpha \setminus^{\alpha'} s^{\downarrow}\}\!\} \\ (s \cdot t)^{\downarrow} & \triangleq & s^{\downarrow} \cdot t^{\downarrow} \end{array}$$

Lemma A.2. *Let $o \in \mathbb{O}_{\Lambda M}$, $u \in \mathbb{T}_{\Lambda M}$ and s be a stack. Then, $(o\{x \setminus u\})^{\downarrow} = o^{\downarrow}\{x \setminus u^{\downarrow}\}$ and $(o\{\!\{\alpha \setminus^{\alpha'} s\}\!\})^{\downarrow} = o^{\downarrow}\{\!\{\alpha \setminus^{\alpha'} s^{\downarrow}\}\!\}$.*

Proof. Both statements are by induction on o. □

Lemma A.3. *Let $o, o' \in \mathbb{O}_{\lambda\mu}$ and $u, u' \in \mathbb{T}_{\lambda\mu}$ and s, s' be stacks such that $o \to_{\lambda\mu} o'$, $u \to_{\lambda\mu} u'$ and $s \to_{\lambda\mu} s'$. Then,*

(1) $o\{x \setminus u\} \to_{\lambda\mu} o'\{x \setminus u\}$.
(2) $o\{x \setminus u\} \twoheadrightarrow_{\lambda\mu} o\{x \setminus u'\}$.
(3) $o\{\!\{\alpha \setminus^{\alpha'} s\}\!\} \to_{\lambda\mu} o'\{\!\{\alpha \setminus^{\alpha'} s\}\!\}$.
(4) $o\{\!\{\alpha \setminus^{\alpha'} s\}\!\} \twoheadrightarrow_{\lambda\mu} o\{\!\{\alpha \setminus^{\alpha'} s'\}\!\}$.

Proof. Items (1) and (3) are by induction on $o \to_{\lambda\mu} o'$, while items (2) and (4) are by induction on o. □

Last, to apply the interpretation method we need to relate the relations $\to_{\Lambda M}$ and $\to_{\lambda\mu}$ by means the projection function $\_^{\downarrow}$.

Lemma A.4.

(1) *Let $o \in \mathbb{O}_{\Lambda M}$. Then, $o \twoheadrightarrow_{\Lambda M} o^{\downarrow}$.*
(2) *Let $o, o' \in \mathbb{O}_{\lambda\mu}$. If $o \to_{\lambda\mu} o'$, then $o \twoheadrightarrow_{\Lambda M} o'$.*
(3) *Let $o, o' \in \mathbb{O}_{\Lambda M}$. If $o \to_{\Lambda M} o'$, then $o^{\downarrow} \twoheadrightarrow_{\lambda\mu} o'^{\downarrow}$.*

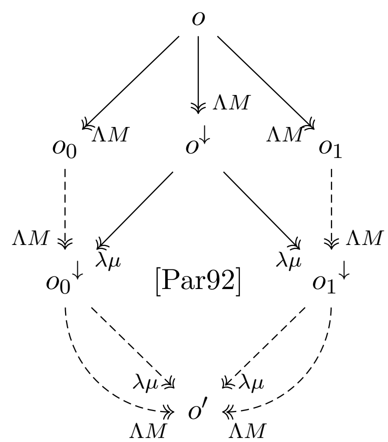
Proof.

(1) By induction on o.
(2) By induction on $o \to_{\lambda\mu} o'$.
(3) By induction on $o \to_{\Lambda M} o'$ using Lemma A.2, and Lemma A.3. □

Confluence of $\to_{\Lambda M}$ is a consequence of Lemma A.4 and confluence of $\to_{\lambda\mu}$ [Par92].

Theorem 4.3. *The reduction relation $\to_{\Lambda M}$ is confluent (CR).*

Proof. By the interpretation method, using confluence of $\to_{\lambda\mu}$ and Lemma A.4:



where $o \twoheadrightarrow_{\Lambda M} o_0$ and $o \twoheadrightarrow_{\Lambda M} o_1$ are the hypothesis of the theorem. The three vertical reductions are justified by Lemma A.4 (1), since $p \twoheadrightarrow_{\Lambda M} p^{\downarrow}$ for all $o \in \mathbb{O}_{\Lambda M}$. The reductions $o^{\downarrow} \twoheadrightarrow_{\lambda\mu} {o_0}^{\downarrow}$ and $o^{\downarrow} \twoheadrightarrow_{\lambda\mu} {o_1}^{\downarrow}$ come from Lemma A.4 (3). The diagram is closed by [Par92] thus obtaining ${o_0}^{\downarrow} \twoheadrightarrow_{\Lambda M} o'$ and ${o_1}^{\downarrow} \twoheadrightarrow_{\Lambda M} o'$ by Lemma A.4 (2). □

APPENDIX B. STRONG NORMALISATION OF PLAIN COMPUTATION

To show that plain computation is strongly normalising we define a measure over objects of the ΛM-calculus. It is worth noticing that using the standard size of an object (*i.e.* counting all its constructors) is not enough since it does not strictly decrease under computation due to the following remarks:

(1) Rule `dM` discards an application while introducing a new explicit replacement, thus preserving the number of constructors in the object.
(2) Rule `N` discards a linear explicit replacement with a stack of n elements, replacing it with n applications. The number of stack constructors in a stack of n elements turns out to be $n-1$ which, together with the discarded explicit replacement, compensates the n introduced applications.
(3) Rule `C` discards a linear explicit replacement by combining it with another one, appending their respective stacks. This introduces a new stack constructor, preserving the total number of constructors in the object.

The first remark suggests that the application constructor should have more weight than the replacement constructor to guarantee normalisation by means of a polynomial interpretation. However, the second remark suggests exactly the opposite. On another note, the third remark requires explicit replacements to have more weight than stacks.

Under these considerations we define the following measure over objects of the ΛM-calculus which turns out to be decreasing w.r.t. reduction $\to_{\mathcal{P}}$.

$$
\begin{array}{lcl@{\qquad}lcl}
(\!|x|\!) & \triangleq & 3 & (\!|[\alpha]\,t|\!) & \triangleq & (\!|t|\!) \\
(\!|t\,u|\!) & \triangleq & (\!|t|\!) * (\!|u|\!) & (\!|c[\![\alpha \setminus^{\alpha'} s]\!]|\!) & \triangleq & (\!|c|\!) * (\!|s|\!) + 1 \\
(\!|\lambda x.t|\!) & \triangleq & (\!|t|\!) & & & \\
(\!|\mu\alpha.c|\!) & \triangleq & (\!|c|\!) + 1 & (\!|t \cdot s|\!) & \triangleq & (\!|t|\!) * (\!|s|\!) \\
(\!|t[x \setminus u]|\!) & \triangleq & (\!|t|\!) + (\!|u|\!) & & &
\end{array}
$$

Notice that $(\!|o|\!) \geq 3$ for every $o \in \mathbb{O}_{\Lambda M}$. We also have $(\!|o|\!) + (\!|o'|\!) < (\!|o|\!) * (\!|o'|\!)$.

Lemma B.1. *For every* $\mathtt{O}$*, there is* $f_{\mathtt{O}} : x \mapsto a * x + b$ *with* $a > 0$ *and* $b \geq 0$*, such that for every* o*, one has* $(\!|\mathtt{O}\langle o\rangle|\!) = f_{\mathtt{O}}((\!|o|\!))$*.*

Proof. By induction on $\mathtt{O}$.

- For $\mathtt{O} = \Box$ and $\mathtt{O} = \boxdot$ we take $f_{\mathtt{O}}(x) = 1 * x + 0$.
- For $\mathtt{O} = \mathtt{T}[x \setminus t]$ and $\mathtt{O} = t[x \setminus \mathtt{T}]$, we take $f_{\mathtt{O}}(x) = a' * x + (b' + (\!|t|\!))$, where $f_{\mathtt{T}}(x) = a' * x + b'$.
- For $\mathtt{O} = \mathtt{T}\, t$ and $\mathtt{O} = t\, \mathtt{T}$ we take $f_{\mathtt{O}}(x) = (a' * (\!|t|\!)) * x + (b' * (\!|t|\!))$, where $f_{\mathtt{T}}(x) = a' * x + b'$.
- For $\mathtt{O} = \lambda x.\mathtt{T}$ and $\mathtt{O} = [\alpha]\, \mathtt{T}$ we take $f_{\mathtt{O}}(x) = a' * x + b'$, where $f_{\mathtt{T}}(x) = a' * x + b'$.
- For $\mathtt{O} = \mu\alpha.\mathtt{C}$ we take $f_{\mathtt{O}}(x) = a' * x + (b' + 1)$, where $f_{\mathtt{C}}(x) = a' * x + b'$.
- For $\mathtt{O} = \mathtt{C}[\![\alpha \setminus^{\alpha'} s]\!]$ we take $f_{\mathtt{O}}(x) = (a' * (\!|s|\!)) * x + (b' * (\!|s|\!) + 1)$, where $f_{\mathtt{C}}(x) = a' * x + b'$.
- For $\mathtt{O} = c[\![\alpha \setminus^{\alpha'} \mathtt{S}]\!]$ we take $f_{\mathtt{O}}(x) = (a' * (\!|c|\!)) * x + (b' * (\!|c|\!) + 1)$, where $f_{\mathtt{S}}(x) = a' * x + b'$.
- For $\mathtt{O} = \mathtt{T} \cdot s$ we take $f_{\mathtt{O}}(x) = (a' * (\!|s|\!)) * x + (b' * (\!|s|\!) + 1)$, where $f_{\mathtt{T}}(x) = a' * x + b'$.
- For $\mathtt{O} = t \cdot \mathtt{S}$ we take $f_{\mathtt{O}}(x) = (a' * (\!|t|\!)) * x + (b' * (\!|t|\!) + 1)$, where $f_{\mathtt{S}}(x) = a' * x + b'$. ☐

Theorem 5.2. *The relation* $\to_{\mathcal{P}}$ *is terminating.*

Proof. We prove $o \to_{\mathcal{P}} p$ implies $(\!|o|\!) > (\!|p|\!)$ by induction on the relation $\to_{\mathcal{P}}$. We first analyse all the base cases:

- $o = \mathtt{L}\langle \lambda x.t\rangle\, u \mapsto_{\mathtt{dB}} \mathtt{L}\langle t[x \setminus u]\rangle = p$, where $\mathsf{fc}(u, \mathtt{L})$. Then Lemma B.1 gives $f_{\mathtt{L}}(x) = a * x + b$, with $a > 0$ and $b \geq 0$. We conclude by

$$\begin{array}{rcl} (\!|o|\!) & = & (a * (\!|t|\!) + b) * (\!|u|\!) \\ & = & a * (\!|t|\!) * (\!|u|\!) + b * (\!|u|\!) \\ & > & a * ((\!|t|\!) + (\!|u|\!)) + b * (\!|u|\!) \\ & \geq & a * ((\!|t|\!) + (\!|u|\!)) + b \\ & = & (\!|p|\!) \end{array}$$

- $o = \mathtt{L}\langle \mu\alpha.c\rangle\, u \mapsto_{\mathtt{dM}} \mathtt{L}\langle \mu\alpha'.c[\![\alpha \setminus^{\alpha'} u]\!]\rangle = p$, where α' is fresh. Then Lemma B.1 gives $f_{\mathtt{L}}(x) = a * x + b$, with $a > 0$ and $b \geq 0$. We conclude by

$$\begin{array}{rcl} (\!|o|\!) & = & (a * ((\!|c|\!) + 1) + b) * (\!|u|\!) \\ & = & a * (\!|c|\!) * (\!|u|\!) + (\!|u|\!) * a + (\!|u|\!) * b \\ & > & a * (\!|c|\!) * (\!|u|\!) + 2 * a + b \\ & = & a * (((\!|c|\!) * (\!|u|\!) + 1) + 1) + b \\ & = & (\!|p|\!) \end{array}$$

- $o = \mathtt{LCC}\langle [\alpha]\, t\rangle[\![\alpha \setminus^{\alpha'} s]\!] \mapsto_{\mathtt{N}} \mathtt{LCC}\langle [\alpha']\, t :: s\rangle = p$, where $\alpha \notin \mathtt{LCC}$ and $\alpha \notin t$. Assume $s = u_0 \cdot \ldots \cdot u_n$. Lemma B.1 gives $f_{\mathtt{LCC}}(x) = a * x + b$, with $a > 0$ and $b \geq 0$. We conclude by

$$\begin{array}{rcl} (\!|o|\!) & = & (a * (\!|t|\!) + b) * (\!|u_0|\!) * \ldots * (\!|u_n|\!) + 1 \\ & = & a * (\!|t|\!) * (\!|u_0|\!) * \ldots * (\!|u_n|\!) + b * (\!|u_0|\!) * \ldots * (\!|u_n|\!) + 1 \\ & > & a * (\!|t|\!) * (\!|u_0|\!) * \ldots * (\!|u_n|\!) + b \\ & = & (\!|p|\!) \end{array}$$

- $o = \mathtt{LCC}\langle c'[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!] \mapsto_{\mathtt{C}} \mathtt{LCC}\langle c'[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle = p$, where $\alpha \notin \mathtt{LCC}$, $\alpha \notin c'$ and $\alpha \notin s'$. Lemma B.1 gives $f_{\mathtt{LCC}}(x) = a * x + b$, with $a > 0$ and $b \geq 0$. We conclude by

$$\begin{array}{rcl} (\!|o|\!) & = & (a * ((\!|c'|\!) * (\!|s'|\!) + 1) + b) * (\!|s|\!) + 1 \\ & = & a * (\!|c'|\!) * (\!|s'|\!) * (\!|s|\!) + (a+b) * (\!|s|\!) + 1 \\ & > & a * (\!|c'|\!) * (\!|s'|\!) * (\!|s|\!) + a + b \\ & = & a * ((\!|c'|\!) * (\!|s'|\!) * (\!|s|\!) + 1) + b \\ & = & (\!|p|\!) \end{array}$$

For every inductive case of the form $o = \mathtt{O}\langle o'\rangle \to_{\mathcal{P}} \mathtt{O}\langle p'\rangle = p$ where $o' \to_{\mathcal{P}} p'$ is a base reduction step, we get $(\!|o'|\!) > (\!|p'|\!)$ by the *i.h.* We then use Lemma B.1 to get $(\!|o|\!) = f_{\mathtt{O}}((\!|o'|\!))$ and $(\!|p|\!) = f_{\mathtt{O}}((\!|p'|\!))$. We conclude $(\!|o|\!) > (\!|p|\!)$ since $f_{\mathtt{O}}$ is clearly strictly monotone by construction. □

Theorem 5.3. *The relation $\to_{\mathcal{P}}$ has the diamond property and hence it is confluent.*

Proof. We first remark that the rules dB, dM, N and C do not duplicate any subterm. Thus, any trivial one-step divergence between these rules can be easily closed in one step as well. Then, the only cases left to be considered are the critical pairs between them. There are only two such cases:

(1) N–C. We have $o = \mathtt{LCC}_2\langle \mathtt{LCC}_1\langle[\delta]\, t\rangle[\![\delta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!]$ with the conditions $\alpha \notin \mathtt{LCC}_2$, $\alpha \notin \mathtt{LCC}_1\langle[\delta]\, t\rangle$, $\alpha \notin s$, $\mathsf{fc}(\alpha', \mathtt{LCC}_2)$, $\mathsf{fc}(s, \mathtt{LCC}_2)$ given by rule C, and the conditions $\delta \notin t$, $\delta \notin \mathtt{LCC}_1$, $\mathsf{fc}(\alpha, \mathtt{LCC}_1)$ and $\mathsf{fc}(s', \mathtt{LCC}_1)$ given by rule N. We conclude since $(t :: s') :: s = t :: (s' \cdot s)$, thus obtaining the diagram:

$$\begin{array}{ccc} \mathtt{LCC}_2\langle \mathtt{LCC}_1\langle[\delta]\, t\rangle[\![\delta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \to_{\mathtt{N}} & \mathtt{LCC}_2\langle \mathtt{LCC}_1\langle[\alpha]\, t :: s'\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!] \\ {\scriptstyle\mathtt{C}}\downarrow & & {\scriptstyle\mathtt{N}}\downarrow \\ \mathtt{LCC}_2\langle \mathtt{LCC}_1\langle[\delta]\, t\rangle[\![\delta \setminus^{\alpha'} s' \cdot s]\!]\rangle & \to_{\mathtt{N}} & \mathtt{LCC}_2\langle \mathtt{LCC}_1\langle[\alpha']\,(t :: s') :: s\rangle\rangle \end{array}$$

(2) C–C. Then we have $o = \mathtt{LCC}_2\langle \mathtt{LCC}_1\langle c[\![\gamma \setminus^{\delta} s'']\!]\rangle[\![\delta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!]$ with the conditions $\alpha \notin \mathtt{LCC}_2$, $\alpha \notin \mathtt{LCC}_1\langle c[\![\gamma \setminus^{\delta} s'']\!]\rangle$, $\alpha \notin s$, $\mathsf{fc}(\alpha', \mathtt{LCC}_2)$, $\mathsf{fc}(s, \mathtt{LCC}_2)$ due to the outermost application of the rule, and the conditions $\delta \notin c$, $\delta \notin \mathtt{LCC}_1$, $\delta \notin s'$, $\mathsf{fc}(\alpha, \mathtt{LCC}_1)$ and $\mathsf{fc}(s', \mathtt{LCC}_1)$ due to the innermost one. We conclude since $s'' \cdot (s' \cdot s) = (s'' \cdot s') \cdot s$, thus obtaining the following diagram:

$$\begin{array}{ccc} \mathtt{LCC}_2\langle \mathtt{LCC}_1\langle c[\![\gamma \setminus^{\delta} s'']\!]\rangle[\![\delta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \to_{\mathtt{C}} & \mathtt{LCC}_2\langle \mathtt{LCC}_1\langle c[\![\gamma \setminus^{\alpha} s'' \cdot s']\!]\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!] \\ {\scriptstyle\mathtt{C}}\downarrow & & {\scriptstyle\mathtt{C}}\downarrow \\ \mathtt{LCC}_2\langle \mathtt{LCC}_1\langle c[\![\gamma \setminus^{\delta} s'']\!]\rangle[\![\delta \setminus^{\alpha'} s' \cdot s]\!]\rangle & \to_{\mathtt{C}} & \mathtt{LCC}_2\langle \mathtt{LCC}_1\langle c[\![\gamma \setminus^{\alpha'} s'' \cdot s' \cdot s]\!]\rangle\rangle \end{array}$$

□

Appendix C. Structural Equivalence for the ΛM-Calculus

To prove Lemma 5.6 we introduce two auxiliary results about contexts LTT and LCC.

Lemma C.1.

(1) *Suppose $u = \mathtt{L}\langle \mathtt{LTT}\langle t\rangle\rangle$ with $\mathsf{bv}(\mathtt{L}) \notin \mathtt{LTT}$ and $\mathsf{fc}(\mathtt{L}, \mathtt{LTT})$. If $u \to_{\mathcal{P}} u'$, then there exists $\mathtt{LTT}'$, $\mathtt{L}'$ and t' such that $u' = \mathtt{L}'\langle \mathtt{LTT}'\langle t'\rangle\rangle$. Moreover $v = \mathtt{LTT}\langle \mathtt{L}\langle t\rangle\rangle \to_{\mathcal{P}} \mathtt{LTT}'\langle \mathtt{L}'\langle t'\rangle\rangle = v'$. In a diagram:*

$$\begin{array}{ccc} u = \mathtt{L}\langle \mathtt{LTT}\langle t\rangle\rangle & & \mathtt{LTT}\langle \mathtt{L}\langle t\rangle\rangle = v \\ \mathcal{P}\downarrow & \textit{implies} & \mathcal{P}\downarrow \\ u' = \mathtt{L}'\langle \mathtt{LTT}'\langle t'\rangle\rangle & & \mathtt{LTT}'\langle \mathtt{L}'\langle t'\rangle\rangle = v' \end{array}$$

(2) *Suppose* $v = \mathtt{LTT}\langle \mathtt{L}\langle t\rangle\rangle$ *with* $\mathsf{bv}(\mathtt{L}) \notin \mathtt{LTT}$ *and* $\mathsf{fc}(\mathtt{L}, \mathtt{LTT})$. *If* $v \to_{\mathcal{P}} v'$, *then there exists* $\mathtt{LTT}'$, $\mathtt{L}'$ *and* t' *such that* $v' = \mathtt{LTT}'\langle \mathtt{L}'\langle t'\rangle\rangle$. *Moreover,* $u = \mathtt{L}\langle \mathtt{LTT}\langle t\rangle\rangle \to_{\mathcal{P}} \mathtt{L}'\langle \mathtt{LTT}'\langle t'\rangle\rangle = u'$. *On a diagram,*

$$\begin{array}{ccc} v = \mathtt{LTT}\langle \mathtt{L}\langle t\rangle\rangle & & \mathtt{L}\langle \mathtt{LTT}\langle t\rangle\rangle = u \\ \mathcal{P}\downarrow & \textit{implies} & \mathcal{P}\downarrow \\ v' = \mathtt{LTT}'\langle \mathtt{L}'\langle t'\rangle\rangle & & \mathtt{L}'\langle \mathtt{LTT}'\langle t'\rangle\rangle = u' \end{array}$$

Proof. We address the first item, the second one is similar. The possible overlap between the $\mathcal{P}$-step and $\mathtt{L}\langle \mathtt{LTT}\langle t\rangle\rangle$ can be broken down as follows:

- The step is entirely within t, *i.e.* $t \to_{\mathcal{P}} t'$. Then it suffices to take $\mathtt{L}' = \mathtt{L}$ and $\mathtt{LTT}' = \mathtt{LTT}$.
- The step is entirely within $\mathtt{L}$, *i.e.* $\mathtt{L} \to_{\mathcal{P}} \mathtt{L}'$. Then we take $t' = t$ and $\mathtt{LTT}' = \mathtt{LTT}$.
- The step is entirely within $\mathtt{LTT}$, *i.e.* $\mathtt{LTT} \to_{\mathcal{P}} \mathtt{LTT}'$. Then we take $t' = t$ and $\mathtt{L}' = \mathtt{L}$.
- The step overlaps with $\mathtt{LTT}$. There are four cases according to the reduction rule applied:
  (1) $\mathtt{dB}$. There are two further cases.
    (a) It overlaps with t. Then, $t = \mathtt{L}_1\langle \lambda x.v\rangle$, $\mathtt{LTT} = \mathtt{LTT}_2\langle \mathtt{L}_2\, u\rangle$ and the LHS of the rule $\mathtt{dB}$ is $\mathtt{L}_2\langle \mathtt{L}_1\langle \lambda x.v\rangle\rangle\, u$. We conclude by setting $t' = \mathtt{L}_1\langle v[x \setminus u]\rangle$, $\mathtt{L}' = \mathtt{L}$ and $\mathtt{LTT}' = \mathtt{LTT}_2\langle \mathtt{L}_2\rangle$:

$$\begin{array}{cc} \mathtt{L}\langle \mathtt{LTT}\langle t\rangle\rangle & \mathtt{LTT}_2\langle \mathtt{L}_2\langle \mathtt{L}\langle \mathtt{L}_1\langle \lambda x.v\rangle\rangle\rangle\, u\rangle \\ \mathtt{dB}\downarrow & \mathtt{dB}\downarrow \\ \mathtt{L}\langle \mathtt{LTT}_2\langle \mathtt{L}_2\langle t'\rangle\rangle\rangle & \mathtt{LTT}_2\langle \mathtt{L}_2\langle \mathtt{L}\langle \mathtt{L}_1\langle v[x \setminus u]\rangle\rangle\rangle\rangle \end{array}$$

    (b) It does not overlap with t. Then, $\mathtt{LTT} = \mathtt{LTT}_2\langle \mathtt{L}_2\langle \lambda x.\mathtt{LTT}_1\rangle\, u\rangle$. We conclude with $t' = t$, $\mathtt{L}' = \mathtt{L}$ and $\mathtt{LTT}' = \mathtt{LTT}_2\langle \mathtt{L}_2\langle \mathtt{LTT}_1[x \setminus u]\rangle\rangle$:

$$\begin{array}{cc} \mathtt{L}\langle \mathtt{LTT}\langle t\rangle\rangle & \mathtt{LTT}_2\langle \mathtt{L}_2\langle \lambda x.\mathtt{LTT}_1\langle \mathtt{L}\langle t\rangle\rangle\rangle\, u\rangle \\ \mathtt{dB}\downarrow & \mathtt{dB}\downarrow \\ \mathtt{L}\langle \mathtt{LTT}'\langle t\rangle\rangle & \mathtt{LTT}_2\langle \mathtt{L}_2\langle \mathtt{LTT}_1\langle \mathtt{L}\langle t\rangle\rangle[x \setminus u]\rangle\rangle \end{array}$$

  (2) $\mathtt{dM}$. There are two further cases.
    (a) It overlaps with t. Then, $t = \mathtt{L}_1\langle \mu\alpha.c\rangle$, $\mathtt{LTT} = \mathtt{LTT}_2\langle \mathtt{L}_2\, u\rangle$ and the LHS of the step $\mathtt{dM}$ is $\mathtt{L}_2\langle \mathtt{L}_1\langle \mu\alpha.c\rangle\rangle\, u$. We conclude with $t' = \mathtt{L}_1\langle \mu\alpha'.c[\![\alpha \setminus^{\alpha'} u]\!]\rangle$, $\mathtt{L}' = \mathtt{L}$ and $\mathtt{LTT}' = \mathtt{LTT}_2\langle \mathtt{L}_2\rangle$:

$$\begin{array}{cc} \mathtt{L}\langle \mathtt{LTT}\langle t\rangle\rangle & \mathtt{LTT}_2\langle \mathtt{L}_2\langle \mathtt{L}\langle \mathtt{L}_1\langle \mu\alpha.c\rangle\rangle\rangle\, u\rangle \\ \mathtt{dM}\downarrow & \mathtt{dM}\downarrow \\ \mathtt{L}\langle \mathtt{LTT}'\langle t'\rangle\rangle & \mathtt{LTT}_2\langle \mathtt{L}_2\langle \mathtt{L}\langle \mathtt{L}_1\langle \mu\alpha'.c[\![\alpha \setminus^{\alpha'} u]\!]\rangle\rangle\rangle\rangle \end{array}$$

(b) It does not overlap with t. Then, $\mathtt{LTT} = \mathtt{LTT}_2\langle \mathtt{L}_2\langle \mu\alpha.\mathtt{LCT}\rangle\, u\rangle$. We conclude with $t' = t$, $\mathtt{L}' = \mathtt{L}$ and $\mathtt{LTT}' = \mathtt{LTT}_2\langle \mathtt{L}_2\langle \mu\alpha'.\mathtt{LCT}[\![\alpha \setminus^{\alpha'} u]\!]\rangle\rangle$:

$$\begin{array}{ccc}
\mathtt{L}\langle \mathtt{LTT}\langle t\rangle\rangle & & \mathtt{LTT}_2\langle \mathtt{L}_2\langle \mu\alpha.\mathtt{LCT}\langle \mathtt{L}\langle t\rangle\rangle\, u\rangle \\
\downarrow_{\mathtt{dM}} & & \downarrow_{\mathtt{dM}} \\
\mathtt{L}\langle \mathtt{LTT}'\langle t\rangle\rangle & & \mathtt{LTT}_2\langle \mathtt{L}_2\langle \mu\alpha'.\mathtt{LCT}\langle \mathtt{L}\langle t\rangle\rangle[\![\alpha \setminus^{\alpha'} u]\!]\rangle\rangle
\end{array}$$

(3) $\mathtt{N}$. There are two further cases.

(a) It overlaps with t. Then, $t = \mathtt{LTC}_1\langle [\alpha]\, u\rangle$, $\mathtt{LTT} = \mathtt{LTC}_2\langle \mathtt{LCT}[\![\alpha \setminus^{\alpha'} s]\!]\rangle$ and the LHS of the step $\mathtt{N}$ is $\mathtt{LCT}\langle \mathtt{LTC}_1\langle [\alpha]\, u\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!]$. We conclude by setting $t' = \mathtt{LTC}_1\langle [\alpha']\, u :: s\rangle$, $\mathtt{L}' = \mathtt{L}$ and $\mathtt{LTT}' = \mathtt{LTT}_2\langle \mu\delta.\mathtt{LCC}_2\langle \mathtt{LCT}\rangle\rangle$:

$$\begin{array}{ccc}
\mathtt{L}\langle \mathtt{LTT}\langle t\rangle\rangle & & \mathtt{LTC}_2\langle \mathtt{LCT}\langle \mathtt{L}\langle \mathtt{LTC}_1\langle [\alpha]\, u\rangle\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle \\
\downarrow_{\mathtt{N}} & & \downarrow_{\mathtt{N}} \\
\mathtt{L}\langle \mathtt{LTT}'\langle t'\rangle\rangle & & \mathtt{LTC}_2\langle \mathtt{LCT}\langle \mathtt{L}\langle \mathtt{LTC}_1\langle [\alpha']\, u :: s\rangle\rangle\rangle\rangle
\end{array}$$

(b) It does not overlap with t. Then, $\mathtt{LTT} = \mathtt{LTT}_2\langle \mu\gamma.\mathtt{LCC}_2\langle \mathtt{LCC}\langle [\alpha]\, \mathtt{LCT}\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle\rangle$. We conclude with $t' = t$, $\mathtt{L}' = \mathtt{L}$ and $\mathtt{LTT}' = \mathtt{LTT}_2\langle \mu\gamma.\mathtt{LCC}_2\langle \mathtt{LCC}\langle [\alpha']\, \mathtt{LCT} :: s\rangle\rangle\rangle$:

$$\begin{array}{ccc}
\mathtt{L}\langle \mathtt{LTT}\langle t\rangle\rangle & & \mathtt{LTT}_2\langle \mu\gamma.\mathtt{LCC}_2\langle \mathtt{LCC}\langle [\alpha]\, \mathtt{LCT}\langle \mathtt{L}\langle t\rangle\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle\rangle \\
\downarrow_{\mathtt{N}} & & \downarrow_{\mathtt{N}} \\
\mathtt{L}\langle \mathtt{LTT}'\langle t\rangle\rangle & & \mathtt{LTT}_2\langle \mu\gamma.\mathtt{LCC}_2\langle \mathtt{LCC}\langle [\alpha']\, \mathtt{LCT}\langle \mathtt{L}\langle t\rangle\rangle :: s\rangle\rangle\rangle
\end{array}$$

(4) $\mathtt{C}$. There are two further cases.

(a) It overlaps with t. Then, $t = \mathtt{LTC}_1\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle$, $\mathtt{LTT} = \mathtt{LTC}_2\langle \mathtt{LCT}[\![\alpha \setminus^{\alpha'} s]\!]\rangle$ and the LHS of the step $\mathtt{C}$ is $\mathtt{LCT}\langle \mathtt{LTC}_1\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!]$. We conclude by setting $t' = \mathtt{LTT}_1\langle \mu\gamma.\mathtt{LCC}_1\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle\rangle$, $\mathtt{L}' = \mathtt{L}$ and $\mathtt{LTT}' = \mathtt{LTC}_2\langle \mathtt{LCT}\rangle$:

$$\begin{array}{ccc}
\mathtt{L}\langle \mathtt{LTT}\langle t\rangle\rangle & & \mathtt{LTC}_2\langle \mathtt{LCT}\langle \mathtt{L}\langle \mathtt{LTC}_1\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle \\
\downarrow_{\mathtt{C}} & & \downarrow_{\mathtt{C}} \\
\mathtt{L}\langle \mathtt{LTT}'\langle t'\rangle\rangle & & \mathtt{LTC}_2\langle \mathtt{LCT}\langle \mathtt{L}\langle \mathtt{LTC}_1\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle\rangle\rangle\rangle
\end{array}$$

(b) It does not overlap with t. Then, $\mathtt{LTT} = \mathtt{LTT}_2\langle \mu\gamma.\mathtt{LCC}_2\langle \mathtt{LCC}\langle \mathtt{LCT}[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle\rangle$. We conclude with $t' = t$, $\mathtt{L}' = \mathtt{L}$ and $\mathtt{LTT}' = \mathtt{LTT}_2\langle \mu\gamma.\mathtt{LCC}_2\langle \mathtt{LCC}\langle \mathtt{LCT}[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle\rangle\rangle$:

$$\begin{array}{ccc}
\mathtt{L}\langle \mathtt{LTT}\langle t\rangle\rangle & & \mathtt{LTT}_2\langle \mu\gamma.\mathtt{LCC}_2\langle \mathtt{LCC}\langle \mathtt{LCT}\langle \mathtt{L}\langle t\rangle\rangle[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle\rangle \\
\downarrow_{\mathtt{C}} & & \downarrow_{\mathtt{C}} \\
\mathtt{L}\langle \mathtt{LTT}'\langle t\rangle\rangle & & \mathtt{LTT}_2\langle \mu\gamma.\mathtt{LCC}_2\langle \mathtt{LCC}\langle \mathtt{LCT}\langle \mathtt{L}\langle t\rangle\rangle[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle\rangle\rangle
\end{array}$$

- There are not other cases. □

We recall the definition of Replacement/Renaming Contexts:

$$\textbf{(Repl./Ren. Contexts)} \quad \mathtt{R} \ ::= \ \boxdot \mid \mathtt{R}[\![\alpha \setminus^{\alpha'} s]\!] \mid [\beta]\, \mu\alpha.\mathtt{R}$$

Lemma C.2.

(1) *Suppose* $d = \mathtt{R}\langle\mathtt{LCC}\langle c\rangle\rangle$ *with* $\mathsf{bn}(\mathtt{R}) \notin \mathtt{LCC}$ *and* $\mathsf{fc}(\mathtt{R}, \mathtt{LCC})$. *If* $d \to_{\mathcal{P}} d''$, *then there exists* $\mathtt{R}'$, $\mathtt{LCC}'$ *and* c' *such that* $d'' \twoheadrightarrow_{\mathtt{dM}} \mathtt{R}'\langle\mathtt{LCC}'\langle c'\rangle\rangle$. *Moreover,* $e = \mathtt{LCC}\langle\mathtt{R}\langle c\rangle\rangle \to_{\mathcal{P}} e'' \twoheadrightarrow_{\mathtt{dM}} \mathtt{LCC}'\langle\mathtt{R}'\langle c'\rangle\rangle = e'$. *In a diagram:*

$$\begin{array}{ccc}
d = \mathtt{R}\langle\mathtt{LCC}\langle c\rangle\rangle & & \mathtt{LCC}\langle\mathtt{R}\langle c\rangle\rangle = e \\
\downarrow \mathcal{P} & & \downarrow \mathcal{P} \\
d'' & \textit{implies} & e'' \\
\twoheadrightarrow \mathtt{dM} & & \twoheadrightarrow \mathtt{dM} \\
d' = \mathtt{R}'\langle\mathtt{LCC}'\langle c'\rangle\rangle & & \mathtt{LCC}'\langle\mathtt{R}'\langle c'\rangle\rangle = e'
\end{array}$$

(2) *Suppose* $e = \mathtt{LCC}\langle\mathtt{R}\langle c\rangle\rangle$ *with* $\mathsf{bn}(\mathtt{R}) \notin \mathtt{LCC}$ *and* $\mathsf{fc}(\mathtt{R}, \mathtt{LCC})$. *If* $e \to_{\mathcal{P}} e''$, *then there exists* $\mathtt{R}'$, $\mathtt{LCC}'$ *and* c' *such that* $e'' \twoheadrightarrow_{\mathtt{dM}} \mathtt{LCC}'\langle\mathtt{R}'\langle c'\rangle\rangle$, *Moreover,* $d = \mathtt{R}\langle\mathtt{LCC}\langle c\rangle\rangle \to_{\mathcal{P}} d'' \twoheadrightarrow_{\mathtt{dM}} \mathtt{R}'\langle\mathtt{LCC}'\langle c'\rangle\rangle = d'$. *In a diagram,*

$$\begin{array}{ccc}
e = \mathtt{LCC}\langle\mathtt{R}\langle c\rangle\rangle & & \mathtt{R}\langle\mathtt{LCC}\langle c\rangle\rangle = d \\
\downarrow \mathcal{P} & & \downarrow \mathcal{P} \\
e'' & \textit{implies} & d'' \\
\twoheadrightarrow \mathtt{dM} & & \twoheadrightarrow \mathtt{dM} \\
e' = \mathtt{LCC}'\langle\mathtt{R}'\langle c'\rangle\rangle & & \mathtt{R}'\langle\mathtt{LCC}'\langle c'\rangle\rangle = d'
\end{array}$$

Proof. We focus on the first item, the second being similar. The proof proceeds by analysing all the overlapping cases between the LHS of the $\mathcal{P}$-step and $\mathtt{R}\langle\mathtt{LCC}\langle c\rangle\rangle$:

- The step is completely within c, *i.e.* $c \to_{\mathcal{P}} c'$. Then, it suffices to set $\mathtt{R}' = \mathtt{R}$ and $\mathtt{LCC}' = \mathtt{LCC}$ and the reduction sequences $d'' \twoheadrightarrow_{\mathtt{dM}} d'$ and $e'' \twoheadrightarrow_{\mathtt{dM}} e'$ are empty.
- The step is completely within $\mathtt{R}$, *i.e.* $\mathtt{R} \to_{\mathcal{P}} \mathtt{R}'$. Then, it suffices to set $c' = c$ and $\mathtt{LCC}' = \mathtt{LCC}$ and the reduction sequences $d'' \twoheadrightarrow_{\mathtt{dM}} d'$ and $e'' \twoheadrightarrow_{\mathtt{dM}} e'$ are empty. This case relies on the fact that all $\mathtt{R}$ contexts are also $\mathtt{LCC}$ contexts and that the latter are present in the patterns of the LHSs of rewrite rules defining $\mathcal{P}$.
- The step is completely within $\mathtt{LCC}$, *i.e.* $\mathtt{LCC} \to_{\mathcal{P}} \mathtt{LCC}'$. Then it suffices to set $c' = c$ and $\mathtt{R}' = \mathtt{R}$ and the reduction sequences $d'' \twoheadrightarrow_{\mathtt{dM}} d'$ and $e'' \twoheadrightarrow_{\mathtt{dM}} e'$ are empty.
- The step overlaps with $\mathtt{LCC}$. There are three further cases depending on the reduction step applied and the reduction sequences $d'' \twoheadrightarrow_{\mathtt{dM}} d'$ and $e'' \twoheadrightarrow_{\mathtt{dM}} e'$ are empty:
  (1) $\mathtt{dB}$. Then, $\mathtt{LCC} = \mathtt{LCC}_1\langle[\gamma]\,\mathtt{LTC}_1\langle\mathtt{L}\langle\lambda x.\mathtt{LTC}_2\rangle\,u\rangle\rangle$. We conclude with $c' = c$, $\mathtt{R}' = \mathtt{R}$ and $\mathtt{LCC}' = \mathtt{LCC}_1\langle[\gamma]\,\mathtt{LTC}_1\langle\mathtt{L}\langle\mathtt{LTC}_2[x \setminus u]\rangle\rangle\rangle$:

$$\begin{array}{cc}
\mathtt{R}\langle\mathtt{LCC}\langle c\rangle\rangle & \mathtt{LCC}_1\langle[\gamma]\,\mathtt{LTC}_1\langle\mathtt{L}\langle\lambda x.\mathtt{LTC}_2\langle\mathtt{R}\langle c\rangle\rangle\rangle\,u\rangle\rangle \\
\downarrow \mathtt{dB} & \downarrow \mathtt{dB} \\
\mathtt{R}\langle\mathtt{LCC}'\langle c\rangle\rangle & \mathtt{LCC}_1\langle[\gamma]\,\mathtt{LTC}_1\langle\mathtt{L}\langle\mathtt{LTC}_2\langle\mathtt{R}\langle c\rangle\rangle[x \setminus u]\rangle\rangle\rangle
\end{array}$$

  (2) $\mathtt{dM}$. Then, $\mathtt{LCC} = \mathtt{LCC}_1\langle[\gamma]\,\mathtt{LTC}_1\langle\mathtt{L}\langle\mu\alpha.\mathtt{LTC}_2\rangle\,u\rangle\rangle$. We conclude with $c' = c$, $\mathtt{R}' = \mathtt{R}$ and $\mathtt{LCC}' = \mathtt{LCC}_1\langle[\gamma]\,\mathtt{LTC}_1\langle\mathtt{L}\langle\mu\alpha'.\mathtt{LTC}_2[\![\alpha \setminus^{\alpha'} u]\!]\rangle\rangle\rangle$ and the reduction sequences $d'' \twoheadrightarrow_{\mathtt{dM}} d'$

and $e'' \twoheadrightarrow_{\mathtt{dM}} e'$ are empty:

$$\begin{array}{ccc} \mathtt{R}\langle\mathtt{LCC}\langle c\rangle\rangle & & \mathtt{LCC}_1\langle[\gamma]\,\mathtt{LTC}_1\langle\mathtt{L}\langle\mu\alpha.\mathtt{LTC}_2\langle\mathtt{R}\langle c\rangle\rangle\,u\rangle\rangle \\ \downarrow \mathtt{dM} & & \downarrow \mathtt{dM} \\ \mathtt{R}\langle\mathtt{LCC}'\langle c\rangle\rangle & & \mathtt{LCC}_1\langle[\gamma]\,\mathtt{LTC}_1\langle\mathtt{L}\langle\mu\alpha'.\mathtt{LTC}_2\langle\mathtt{R}\langle c\rangle\rangle[\![\alpha \setminus^{\alpha'} u]\!]\rangle\rangle\rangle \end{array}$$

(3) $\mathtt{N}$. Then there are two further cases.

(a) It overlaps with c. Then we have $c = \mathtt{LCC}_1\langle[\alpha]\,t\rangle$, $\mathtt{LCC} = \mathtt{LCC}_3\langle\mathtt{LCC}_2[\![\alpha \setminus^{\alpha'} s]\!]\rangle$ and the rule $\mathtt{N}$ applies to $\mathtt{LCC}_2\langle\mathtt{LCC}_1\langle[\alpha]\,t\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!]$. We conclude by setting $c' = \mathtt{LCC}_1\langle[\alpha']\,t :: s\rangle$, $\mathtt{R}' = \mathtt{R}$ and $\mathtt{LCC}' = \mathtt{LCC}_3\langle\mathtt{LCC}_2\rangle$ and the reduction sequences $d'' \twoheadrightarrow_{\mathtt{dM}} d'$ and $e'' \twoheadrightarrow_{\mathtt{dM}} e'$ are empty:

$$\begin{array}{ccc} \mathtt{R}\langle\mathtt{LCC}\langle c\rangle\rangle & & \mathtt{LCC}_3\langle\mathtt{LCC}_2\langle\mathtt{R}\langle\mathtt{LCC}_1\langle[\alpha]\,t\rangle\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle \\ \downarrow \mathtt{N} & & \downarrow \mathtt{N} \\ \mathtt{R}\langle\mathtt{LCC}'\langle c'\rangle\rangle & & \mathtt{LCC}_3\langle\mathtt{LCC}_2\langle\mathtt{R}\langle\mathtt{LCC}_1\langle[\alpha']\,t :: s\rangle\rangle\rangle\rangle \end{array}$$

(b) It does not overlap with c. Then, $\mathtt{LCC} = \mathtt{LCC}_3\langle\mathtt{LCC}_2\langle[\alpha]\,\mathtt{LTC}\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle$. We conclude with $c' = c$, $\mathtt{R}' = \mathtt{R}$ and $\mathtt{LCC}' = \mathtt{LCC}_3\langle\mathtt{LCC}_2\langle[\alpha']\,\mathtt{LTC} :: s\rangle\rangle$ and the reduction sequence $e'' \twoheadrightarrow_{\mathtt{dM}} e'$ is empty:

$$\begin{array}{ccc} \mathtt{R}\langle\mathtt{LCC}\langle c\rangle\rangle & & \mathtt{LCC}_3\langle\mathtt{LCC}_2\langle[\alpha]\,\mathtt{LTC}\langle\mathtt{R}\langle c\rangle\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle \\ \downarrow \mathtt{N} & & \downarrow \mathtt{N} \\ \mathtt{R}\langle\mathtt{LCC}'\langle c\rangle\rangle & & \mathtt{LCC}_3\langle\mathtt{LCC}_2\langle[\alpha']\,\mathtt{LTC}\langle\mathtt{R}\langle c\rangle\rangle :: s\rangle\rangle \end{array}$$

(4) $\mathtt{C}$. Then there are two possibilities.

(a) It overlaps with c. Then we have $c = \mathtt{LCC}_1\langle c_1[\![\beta \setminus^{\alpha} s']\!]\rangle$, $\mathtt{LCC} = \mathtt{LCC}_3\langle\mathtt{LCC}_2[\![\alpha \setminus^{\alpha'} s]\!]\rangle$ and the rule $\mathtt{C}$ applies to $\mathtt{LCC}_2\langle\mathtt{LCC}_1\langle c_1[\![\beta \setminus^{\alpha} s']\!]\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!]$. We conclude by setting $c' = \mathtt{LCC}_1\langle c_1[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle$, $\mathtt{R}' = \mathtt{R}$ and $\mathtt{LCC}' = \mathtt{LCC}_3\langle\mathtt{LCC}_2\rangle$ and the reduction sequences $d'' \twoheadrightarrow_{\mathtt{dM}} d'$ and $e'' \twoheadrightarrow_{\mathtt{dM}} e'$ are empty:

$$\begin{array}{ccc} \mathtt{R}\langle\mathtt{LCC}\langle c\rangle\rangle & & \mathtt{LCC}_3\langle\mathtt{LCC}_2\langle\mathtt{R}\langle\mathtt{LCC}_1\langle c_1[\![\beta \setminus^{\alpha} s']\!]\rangle\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle \\ \downarrow \mathtt{C} & & \downarrow \mathtt{C} \\ \mathtt{R}\langle\mathtt{LCC}'\langle c'\rangle\rangle & & \mathtt{LCC}_3\langle\mathtt{LCC}_2\langle\mathtt{R}\langle\mathtt{LCC}_1\langle c_1[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle\rangle\rangle\rangle \end{array}$$

(b) It does not overlap with c. Then, $\mathtt{LCC} = \mathtt{LCC}_3\langle\mathtt{LCC}_2\langle\mathtt{LCC}_1[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle$. We conclude with $c' = c$, $\mathtt{R}' = \mathtt{R}$ and $\mathtt{LCC}' = \mathtt{LCC}_3\langle\mathtt{LCC}_2\langle\mathtt{LCC}_1[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle\rangle$ and the reduction sequences $d'' \twoheadrightarrow_{\mathtt{dM}} d'$ and $e'' \twoheadrightarrow_{\mathtt{dM}} e'$ are empty:

$$\begin{array}{ccc} \mathtt{R}\langle\mathtt{LCC}\langle c\rangle\rangle & & \mathtt{LCC}_3\langle\mathtt{LCC}_2\langle\mathtt{LCC}_1\langle\mathtt{R}\langle c\rangle\rangle[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle \\ \downarrow \mathtt{C} & & \downarrow \mathtt{C} \\ \mathtt{R}\langle\mathtt{LCC}'\langle c\rangle\rangle & & \mathtt{LCC}_3\langle\mathtt{LCC}_2\langle\mathtt{LCC}_1\langle\mathtt{R}\langle c\rangle\rangle[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle\rangle \end{array}$$

- The step overlaps with $\mathtt{R}$. Since the reduction step cannot be $\mathtt{dB}$ nor $\mathtt{dM}$, there are two further cases to consider.

(1) $\mathtt{N}$. Note that it cannot overlap $\mathtt{LCC}$ too. Indeed, if it did then we have the context $\mathtt{R} = \mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{LCC}_1\langle [\alpha]\, \mathtt{LTC}_1\langle \mu\delta.\mathtt{LTC}_2\rangle\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle$. However, this is not allowed by the condition $\mathtt{bn}(\mathtt{R}) \notin \mathtt{LCC}$. There are two cases to consider.

(a) The step overlaps with c. Then $d = \mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{LCC}\langle \mathtt{LCC}_1\langle [\alpha]\, t\rangle\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle$ and $c = \mathtt{LCC}_1\langle [\alpha]\, t\rangle$. We conclude with $c' = \mathtt{LCC}_1\langle [\alpha']\, t :: s\rangle$, $\mathtt{R}' = \mathtt{R}_1\langle \mathtt{R}_2\rangle$ and $\mathtt{LCC}' = \mathtt{LCC}$ and the reduction sequences $d'' \twoheadrightarrow_{\mathtt{dM}} d'$ and $e'' \twoheadrightarrow_{\mathtt{dM}} e'$ are empty:

$$\begin{array}{ccc}
\mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{LCC}\langle \mathtt{LCC}_1\langle [\alpha]\, t\rangle\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle & & \mathtt{LCC}\langle \mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{LCC}_1\langle [\alpha]\, t\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle\rangle \\
\downarrow{\scriptstyle \mathtt{N}} & & \downarrow{\scriptstyle \mathtt{N}} \\
\mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{LCC}\langle \mathtt{LCC}_1\langle [\alpha']\, t :: s\rangle\rangle\rangle\rangle & & \mathtt{LCC}\langle \mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{LCC}_1\langle [\alpha']\, t :: s\rangle\rangle\rangle\rangle
\end{array}$$

(b) The step does not overlap with c. Then $d = \mathtt{R}_1\langle \mathtt{R}_2\langle [\alpha]\, \mu\beta.\mathtt{R}_3\langle \mathtt{LCC}\langle c\rangle\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle$. We assume that $s = u_1 \cdot \ldots \cdot u_n$. We conclude by considering $c' = c$, $\mathtt{R}' = \mathtt{R}_1\langle \mathtt{R}_2\langle [\alpha']\, (\mu\beta_n.\mathtt{R}_3 [\![\beta \setminus^{\beta_1} u_1]\!] \ldots [\![\beta_{n-1} \setminus^{\beta_n} u_n]\!])\rangle\rangle$ and $\mathtt{LCC}' = \mathtt{LCC}$.

$$\begin{array}{ccc}
\mathtt{R}_1\langle \mathtt{R}_2\langle [\alpha]\, \mu\beta.\mathtt{R}_3\langle \mathtt{LCC}\langle c\rangle\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle & & \mathtt{LCC}\langle \mathtt{R}_1\langle \mathtt{R}_2\langle [\alpha]\, \mu\beta.\mathtt{R}_3\langle c\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle\rangle \\
\downarrow{\scriptstyle \mathtt{N}} & & \downarrow{\scriptstyle \mathtt{N}} \\
\mathtt{R}_1\langle \mathtt{R}_2\langle [\alpha']\, (\mu\beta.\mathtt{R}_3\langle \mathtt{LCC}\langle c\rangle\rangle) :: s\rangle\rangle & & \mathtt{LCC}\langle \mathtt{R}_1\langle \mathtt{R}_2\langle [\alpha']\, (\mu\beta.\mathtt{R}_3\langle c\rangle) :: s\rangle\rangle\rangle \\
\twoheaddownarrow{\scriptstyle \mathtt{dM}} & & \twoheaddownarrow{\scriptstyle \mathtt{dM}} \\
d' & & e'
\end{array}$$

where d' and e' are $\mathtt{R}_1\langle \mathtt{R}_2\langle [\alpha']\, (\mu\beta_n.\mathtt{R}_3\langle \mathtt{LCC}\langle c\rangle\rangle [\![\beta \setminus^{\beta_1} u_1]\!] \ldots [\![\beta_{n-1} \setminus^{\beta_n} u_n]\!])\rangle\rangle$ and $e' = \mathtt{LCC}\langle \mathtt{R}_1\langle \mathtt{R}_2\langle [\alpha']\, (\mu\beta_n.\mathtt{R}_3\langle c\rangle [\![\beta \setminus^{\beta_1} u_1]\!] \ldots [\![\beta_{n-1} \setminus^{\beta_n} u_n]\!])\rangle\rangle\rangle$ respectively.

(2) $\mathtt{C}$. Note that it cannot overlap $\mathtt{LCC}$ too. Indeed, if it did then we have the command $d = \mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c\rangle [\![\beta \setminus^{\alpha} s']\!]\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle$. However, this is not allowed by the condition $\mathtt{bn}(\mathtt{R}) \notin \mathtt{LCC}$. There are two cases to consider.

(a) The step overlaps with c. Then $d = \mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{LCC}\langle \mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c'\rangle [\![\beta \setminus^{\alpha} s']\!]\rangle\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle$ and $c = \mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c'\rangle [\![\beta \setminus^{\alpha} s']\!]\rangle$ and the reduction sequences $d'' \twoheadrightarrow_{\mathtt{dM}} d'$ and $e'' \twoheadrightarrow_{\mathtt{dM}} e'$ are empty:

$$\begin{array}{ccc}
\mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{LCC}\langle \mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c'\rangle [\![\beta \setminus^{\alpha} s']\!]\rangle\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle & & \mathtt{LCC}\langle \mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c'\rangle [\![\beta \setminus^{\alpha} s']\!]\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle\rangle \\
\downarrow{\scriptstyle \mathtt{C}} & & \downarrow{\scriptstyle \mathtt{C}} \\
\mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{LCC}\langle \mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c'\rangle [\![\beta \setminus^{\alpha} s' \cdot s]\!]\rangle\rangle\rangle\rangle & & \mathtt{LCC}\langle \mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c'\rangle [\![\beta \setminus^{\alpha} s' \cdot s]\!]\rangle\rangle\rangle\rangle
\end{array}$$

(b) The step does not overlap with c. Then $d = \mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{R}_3\langle \mathtt{R}_4\langle \mathtt{LCC}\langle c\rangle\rangle [\![\beta \setminus^{\alpha} s']\!]\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle$. Then we set $c' = c$, $\mathtt{R}' = \mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{R}_3\langle \mathtt{R}_4 [\![\beta \setminus^{\alpha} s' \cdot s]\!]\rangle\rangle\rangle$ and $\mathtt{LCC}' = \mathtt{LCC}$ and the reduction sequences $d'' \twoheadrightarrow_{\mathtt{dM}} d'$ and $e'' \twoheadrightarrow_{\mathtt{dM}} e'$ are empty:

$$\begin{array}{ccc}
\mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{R}_3\langle \mathtt{R}_4\langle \mathtt{LCC}\langle c\rangle\rangle [\![\beta \setminus^{\alpha} s']\!]\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle & & \mathtt{LCC}\langle \mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{R}_3\langle \mathtt{R}_4\langle c\rangle [\![\beta \setminus^{\alpha} s']\!]\rangle\rangle [\![\alpha \setminus^{\alpha'} s]\!]\rangle\rangle \\
\downarrow{\scriptstyle \mathtt{C}} & & \downarrow{\scriptstyle \mathtt{C}} \\
\mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{R}_3\langle \mathtt{R}_4\langle \mathtt{LCC}\langle c\rangle\rangle [\![\beta \setminus^{\alpha} s' \cdot s]\!]\rangle\rangle\rangle & & \mathtt{LCC}\langle \mathtt{R}_1\langle \mathtt{R}_2\langle \mathtt{R}_3\langle \mathtt{R}_4\langle c\rangle [\![\beta \setminus^{\alpha} s' \cdot s]\!]\rangle\rangle\rangle\rangle
\end{array}$$

- There are no further cases. □

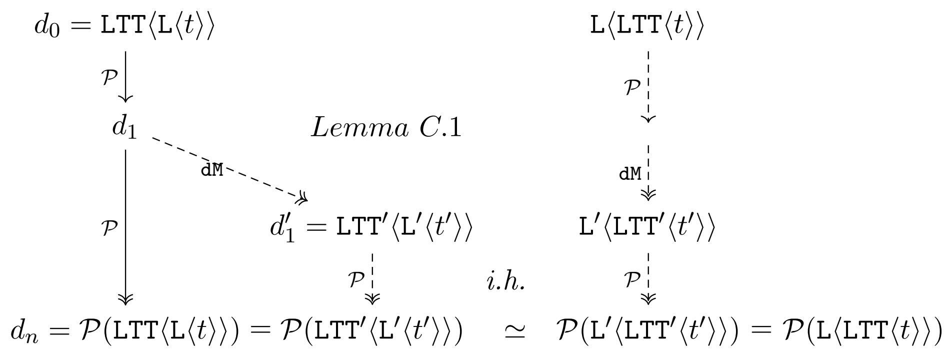
Lemma 5.6.

(1) *Let* $t \in \mathbb{T}_{\Lambda M}$, $\mathtt{L}$ *be a substitution context and* $\mathtt{LTT}$ *a linear context. Then,* $\mathcal{P}(\mathtt{L}\langle\mathtt{LTT}\langle t\rangle\rangle) \simeq \mathcal{P}(\mathtt{LTT}\langle\mathtt{L}\langle t\rangle\rangle)$ *if* $\mathsf{bv}(\mathtt{L}) \notin \mathtt{LTT}$ *and* $\mathsf{fc}(\mathtt{L},\mathtt{LTT})$.

(2) *Let* $c \in \mathbb{C}_{\Lambda M}$, $\mathtt{R}$ *be a repl./ren. context and* $\mathtt{LCC}$ *a linear context. Then,* $\mathcal{P}(\mathtt{R}\langle\mathtt{LCC}\langle c\rangle\rangle) \simeq \mathcal{P}(\mathtt{LCC}\langle\mathtt{R}\langle c\rangle\rangle)$ *if* $\mathsf{bn}(\mathtt{R}) \notin \mathtt{LCC}$ *and* $\mathsf{fc}(\mathtt{R},\mathtt{LCC})$.

Proof.

(1) Let $\mathtt{L}\langle\mathtt{LTT}\langle t\rangle\rangle \in \mathbb{T}_{\Lambda M}$ with $\mathsf{bv}(\mathtt{L}) \notin \mathtt{LTT}$ and $\mathsf{fc}(\mathtt{L},\mathtt{LTT})$. By induction on the length of the longest $\mathcal{P}$ reduction sequence from $\mathtt{L}\langle\mathtt{LTT}\langle t\rangle\rangle$ to its normal form, resorting to Lemma C.1. Suppose $d_0 = \mathtt{LTT}\langle\mathtt{L}\langle t\rangle\rangle \twoheadrightarrow_{\mathcal{P}} \mathcal{P}(\mathtt{LTT}\langle\mathtt{L}\langle t\rangle\rangle) = d_n$ is a longest reduction sequence consisting of n steps. This is depicted on the left, in the figure below. Application of Lemma C.1 will produce the subdiagram at the top of the figure. The $\mathcal{P}$ reduction sequence $d_1 \twoheadrightarrow_{\mathtt{dM}} d_1' \twoheadrightarrow_{\mathcal{P}} d_n$ exists by confluence of $\mathcal{P}$ and, moreover, it has at most $n-1$ steps since the reduction sequence from d_0 was assumed to be a longest reduction. This allows us to apply the *i.h.* to the reduction sequence $d_1' \twoheadrightarrow_{\mathcal{P}} d_n$ (as indicated in the figure) to conclude:



(2) Similar to the previous item but using Lemma C.2. □

Appendix D. Strong Bisimulation Result

Lemma D.1. *Let* $o, o' \in \mathbb{O}_{\Lambda M}$ *such that* $o \simeq_* o'$ *with* $\simeq_*$ *any rule from Figure 5. If* $o = \mathtt{LXC}\langle c\rangle$ *for some* $\mathtt{LXC}$ *and some* c, *with* $\mathtt{X} \in \{\mathtt{T},\mathtt{C}\}$, $\alpha \notin \mathtt{LXC}$, $\mathsf{fc}(\alpha',\mathtt{LXC})$ *and* $\mathsf{fc}(s,\mathtt{LXC})$, *then there exist* $\mathtt{LXC}'$ *and* c' *such that* $o' = \mathtt{LXC}'\langle c'\rangle$ *with* $\alpha \notin \mathtt{LXC}'$, $\mathsf{fc}(\alpha',\mathtt{LXC}')$, $\mathsf{fc}(s,\mathtt{LXC}')$ *and* $\mathcal{P}(\mathtt{LXC}\langle c[\![\alpha \setminus^{\alpha'} s]\!]\rangle) \simeq \mathcal{P}(\mathtt{LXC}'\langle c'[\![\alpha \setminus^{\alpha'} s]\!]\rangle)$.

Proof. By case analysis on $\mathtt{LXC}$. We only illustrate one of the base cases, namely when $\mathtt{LXC} = \boxdot$, the others being similar. In the case that $\mathtt{LXC} = \boxdot$, we must have $o = c$. There are possible rules for commands: $\simeq_{\mathtt{exrepl}}$, $\simeq_{\mathtt{exren}}$, and $\simeq_{\mathtt{ppop}}$. The first two follow by using 5.6 (2), the last one is direct. We provide details on the $\simeq_{\mathtt{ppop}}$ case.

- $\simeq_{\mathtt{ppop}}$. Then, $o = [\gamma']\,\lambda x.\mu\gamma.[\delta']\,\lambda y.\mu\delta.c_0$ and $o' = [\delta']\,\lambda y.\mu\delta.[\gamma']\,\lambda x.\mu\gamma.c_0$ with $\delta \neq \gamma'$ and $\gamma \neq \delta'$. By α-conversion we also assume $x \notin s$, $y \notin s$, $\gamma \neq \alpha'$, $\gamma \notin s$, $\delta \neq \alpha'$ and $\delta \notin s$. Then, there are four possible cases.
  (1) $o[\![\alpha \setminus^{\alpha'} s]\!] \to_{\mathtt{N}} [\alpha']\,(\lambda x.\mu\gamma.[\delta']\,\lambda y.\mu\delta.c_0) :: s$ (*i.e.* $\alpha = \gamma'$, $\alpha \neq \delta'$ and $\gamma' \notin c_0$). Assume $s = u \cdot s'$ (the case $s = u$ is slightly simpler) and consider fresh names γ'', δ''. Then, $\mathcal{P}(o[\![\alpha \setminus^{\alpha'} s]\!]) = \mathcal{P}([\alpha']\,(\mu\gamma''.([\delta']\,\lambda y.\mu\delta.c_0)[\![\gamma \setminus^{\gamma''} s']\!])[x \setminus u])$. Similarly, $\mathcal{P}(o'[\![\alpha \setminus^{\alpha'} s]\!]) =$

$\mathcal{P}([\delta']\,\lambda y.\mu\delta.[\alpha']\,(\mu\gamma''.c_0[\![\gamma\setminus^{\gamma''} s']\!])[x\setminus u])$. In the case $s = u$ we simply avoid explicit replacement. Then, by Lemma 5.6, we have on the one hand

$$\mathcal{P}(o[\![\alpha\setminus^{\alpha'} s]\!]) \simeq \mathcal{P}([\alpha']\,\mu\gamma''.[\delta']\,(\lambda y.\mu\delta.c_0[\![\gamma\setminus^{\gamma''} s']\!])[x\setminus u])$$

and on the other hand

$$\mathcal{P}(o'[\![\alpha\setminus^{\alpha'} s]\!]) \simeq \mathcal{P}([\delta']\,(\lambda y.\mu\delta.[\alpha']\,\mu\gamma''.c_0[\![\gamma\setminus^{\gamma''} s']\!])[x\setminus u])$$

resorting to item (1) of the lemma that positions explicit substitutions and item (2) on explicit replacement. Finally, applying $\simeq_{\texttt{exren}}$ we derive

$$\begin{array}{rcl}
\mathcal{P}(o[\![\alpha\setminus^{\alpha'} s]\!]) & \simeq & \mathcal{P}([\alpha']\,\mu\gamma''.[\delta']\,(\lambda y.\mu\delta.c_0[\![\gamma\setminus^{\gamma''} s']\!])[x\setminus u]) \\
& = & [\alpha']\,\mu\gamma''.[\delta']\,(\lambda y.\mu\delta.\mathcal{P}(c_0[\![\gamma\setminus^{\gamma''} s']\!]))[x\setminus \mathcal{P}(u)] \\
& \simeq_{\texttt{exren}} & [\delta']\,(\lambda y.\mu\delta.[\alpha']\,\mu\gamma''.\mathcal{P}(c_0[\![\gamma\setminus^{\gamma''} s']\!]))[x\setminus \mathcal{P}(u)] \\
& = & \mathcal{P}([\delta']\,(\lambda y.\mu\delta.[\alpha']\,\mu\gamma''.c_0[\![\gamma\setminus^{\gamma''} s']\!])[x\setminus u]) \\
& \simeq & \mathcal{P}(o'[\![\alpha\setminus^{\alpha'} s]\!])
\end{array}$$

We conclude with $\texttt{LXC}' = \boxdot$ and $c' = o'$.

(2) $o[\![\alpha\setminus^{\alpha'} s]\!] \to_{\texttt{N}} [\gamma']\,\lambda x.\mu\gamma.[\alpha']\,(\lambda y.\mu\delta.c_0) :: s$ (*i.e.* $\alpha \neq \gamma'$, $\alpha = \delta'$ and $\delta' \notin c_0$). Similar to the previous case.

(3) $o[\![\alpha\setminus^{\alpha'} s]\!] \to_{\texttt{N,C}} [\gamma']\,\lambda x.\mu\gamma.[\delta']\,\lambda y.\mu\delta.c'_0$ (*i.e.* $\mathsf{fn}_\alpha(o) = 1$, $\alpha \neq \gamma'$ and $\alpha \neq \delta'$). Then,

$$\begin{array}{rcl}
\mathcal{P}(o[\![\alpha\setminus^{\alpha'} s]\!]) & = & [\gamma']\,\lambda x.\mu\gamma.[\delta']\,\lambda y.\mu\delta.\mathcal{P}(c'_0) \\
& \simeq_{\texttt{ppop}} & [\delta']\,\lambda y.\mu\delta.[\gamma']\,\lambda x.\mu\gamma.\mathcal{P}(c'_0) \\
& = & \mathcal{P}(o'[\![\alpha\setminus^{\alpha'} s]\!])
\end{array}$$

We conclude with $\texttt{LXC}' = \boxdot$ and $c' = o'$.

(4) Otherwise, $\mathcal{P}(o[\![\alpha\setminus^{\alpha'} s]\!]) = o[\![\alpha\setminus^{\alpha'} \mathcal{P}(s)]\!] \simeq o'[\![\alpha\setminus^{\alpha'} \mathcal{P}(s)]\!] = \mathcal{P}(o'[\![\alpha\setminus^{\alpha'} s]\!])$, since $o \simeq o'$ implies $\mathcal{P}(o) = o$ and $\mathcal{P}(o') = o'$. We conclude with $\texttt{LXC}' = \boxdot$ and $c' = o'$. □

Lemma D.2. *Let $o, o' \in \mathbb{O}_{\Lambda M}$ such that $o \simeq o'$. Let s be a stack.*

(1) *If $o, o' \in \mathbb{T}_{\Lambda M}$, then $\mathcal{P}(o :: s) \simeq \mathcal{P}(o' :: s)$.*

(2) *If $o, o' \in \mathbb{C}_{\Lambda M}$, then $\mathcal{P}(o[\![\alpha\setminus^{\alpha'} s]\!]) \simeq \mathcal{P}(o'[\![\alpha\setminus^{\alpha'} s]\!])$.*

Proof. We prove both items by simultaneous induction on $o \simeq o'$. The cases where $o \simeq o'$ holds by reflexivity, transitivity or symmetry are straightforward. For congruence, we reason by induction on the context $\texttt{O}$ such that $o = \texttt{O}\langle p\rangle$ and $o' = \texttt{O}\langle p'\rangle$ with $p \simeq_* p'$, where $\simeq_*$ is an axiom in Figure 5.

- $\texttt{O} = \Box$. Then there are only two possible cases which are $\simeq_{\texttt{exsubs}}$ and $\simeq_\theta$. Then we conclude (1) by Lemma 5.6 in the former case, and straightforwardly in the latter.
- $\texttt{O} = \boxdot$. We conclude (2) by Lemma D.1 and Lemma 5.6.
- Cases $\texttt{O} = \texttt{T}\,v$, $\texttt{O} = t\,\texttt{T}$, $\texttt{O} = t[x\setminus \texttt{T}]$ and $\texttt{O} = c[\![\gamma\setminus^{\delta} \texttt{S}]\!]$ are immediate by resorting to the fact that o and o' are already in plain form; while case $\texttt{O} = \lambda x.\texttt{T}$ requires an extra induction on s to conclude.
- $\texttt{O} = \mu\gamma.\texttt{C}$. Here we conclude (1) by resorting to *i.h.* (2).
- $\texttt{O} = \texttt{T}[x\setminus v]$. In this case item (1) follows immediately by *i.h.* (1).
- $\texttt{O} = \texttt{C}[\![\gamma\setminus^{\delta} s']\!]$. In this case we conclude (2) by resorting to *i.h.* (2) and Lemma 5.6.
- $\texttt{O} = [\delta]\,\texttt{T}$. Then $o = [\delta]\,\texttt{T}\langle p\rangle$ and $o' = [\delta]\,\texttt{T}\langle p'\rangle$. Let $t = \texttt{T}\langle p\rangle$ and $t' = \texttt{T}\langle p'\rangle$. We then have $t \simeq t'$ and we consider three possible cases:
  (1) $\alpha = \delta$ and $\alpha \notin t$. Then (2) follows immediately by *i.h.* (1).
  (2) $o[\![\alpha\setminus^{\alpha'} s]\!] \to_{\texttt{N,C}} [\delta]\,u$ (*i.e.* $\alpha \neq \delta$ and $\alpha \in t$). There are two cases:

(a) $u = \mathtt{T}'\langle p\rangle$ for some $\mathtt{T}'$. Similarly, we conclude (2) by resorting to *i.h.* (1).
(b) $u = \mathtt{T}\langle q\rangle$ for some q. Then there is a linear context $\mathtt{LTC}$ such that $\mathtt{LTC} = \mathtt{LTX}\langle\mathtt{LXC}\rangle$, $\mathtt{T} = \mathtt{LTX}$ and $p = \mathtt{LXC}\langle c\rangle$ with $\alpha \neq \delta$, $\alpha \notin \mathtt{LTC}$ and $\alpha \in c$. By α-conversion we assume $\mathsf{fc}(\alpha', \mathtt{LTC})$ and $\mathsf{fc}(s, \mathtt{LTC})$, so that $p[\![\alpha \setminus^{\alpha'} s]\!] \to_{\mathtt{N,C}} q$. Then, by Lemma D.1 with $p \simeq_* p'$ we have $p' = \mathtt{LXC}'\langle c'\rangle$ such that $\alpha \notin \mathtt{LXC}'$, $\mathsf{fc}(\alpha', \mathtt{LXC}')$, $\mathsf{fc}(s, \mathtt{LXC}')$ and $\mathcal{P}(\mathtt{LXC}\langle c[\![\alpha \setminus^{\alpha'} s]\!]\rangle) \simeq \mathcal{P}(\mathtt{LXC}'\langle c'[\![\alpha \setminus^{\alpha'} s]\!]\rangle)$. Finally, $o = [\delta]\,\mathtt{T}\langle p\rangle = [\delta]\,\mathtt{LTX}\langle\mathtt{LXC}\langle c\rangle\rangle$ and $o' = [\delta]\,\mathtt{T}\langle p'\rangle = [\delta]\,\mathtt{LTX}\langle\mathtt{LXC}'\langle c'\rangle\rangle$, and by Lemma 5.6 (2), we conclude (2) as follows

$$\begin{array}{rcl}
\mathcal{P}(o[\![\alpha \setminus^{\alpha'} s]\!]) & \simeq & \mathcal{P}([\delta]\,\mathtt{LTX}\langle\mathtt{LXC}\langle c[\![\alpha \setminus^{\alpha'} s]\!]\rangle\rangle) \\
& = & [\delta]\,\mathtt{LTX}\langle\mathcal{P}(\mathtt{LXC}\langle c[\![\alpha \setminus^{\alpha'} s]\!]\rangle)\rangle \\
& \simeq & [\delta]\,\mathtt{LTX}\langle\mathcal{P}(\mathtt{LXC}'\langle c'[\![\alpha \setminus^{\alpha'} s]\!]\rangle)\rangle \\
& = & \mathcal{P}([\delta]\,\mathtt{LTX}\langle\mathtt{LXC}'\langle c'[\![\alpha \setminus^{\alpha'} s]\!]\rangle\rangle) \\
& \simeq & \mathcal{P}(o'[\![\alpha \setminus^{\alpha'} s]\!])
\end{array}$$

(3) Otherwise, $\mathcal{P}(o[\![\alpha \setminus^{\alpha'} s]\!]) = o[\![\alpha \setminus^{\alpha'} \mathcal{P}(s)]\!] \simeq o'[\![\alpha \setminus^{\alpha'} \mathcal{P}(s)]\!] = \mathcal{P}(o'[\![\alpha \setminus^{\alpha'} s]\!])$, since $o \simeq o'$ implies $\mathcal{P}(o) = o$ and $\mathcal{P}(o') = o'$.

- Cases $\mathtt{O} = \mathtt{T} \cdot s'$ and $\mathtt{O} = t \cdot \mathtt{S}$ do not apply as o and o' are not stacks by hypothesis. □

Lemma 5.7. *Let $o, o' \in \mathbb{O}_{\Lambda M}$. If $o \simeq o'$, then for all context $\mathtt{O}$ of appropriate sort, $\mathcal{P}(\mathtt{O}\langle o\rangle) \simeq \mathcal{P}(\mathtt{O}\langle o'\rangle)$.*

Proof. By induction on $\mathsf{sz}(\mathtt{O})$, where $\mathsf{sz}(\_)$ measures the size of contexts, *i.e.* its number of constructors (excluding $\Box$ and $\boxdot$).

- $\mathtt{O} = \Box$ and $\mathtt{O} = \boxdot$. Immediate since $o \simeq o'$ implies $\mathcal{P}(o) = o$ and $\mathcal{P}(o') = o'$ by definition.
- $\mathtt{O} = \mathtt{T}\,u$. We conclude by the *i.h.* and Lemma D.2 (1).
- $\mathtt{O} = \mathtt{C}[\![\alpha \setminus^{\alpha'} s]\!]$. We conclude by the *i.h.* and Lemma D.2 (2).
- In all the other cases we conclude by the *i.h.* □

Lemma D.3. *Let $o, o' \in \mathbb{O}_{\Lambda M}$ and $u, u' \in \mathbb{T}_{\Lambda M}$ such that $o \simeq o'$ and $u \simeq u'$.*
(1) *Then, $\mathcal{P}(o\{x \setminus u\}) \simeq \mathcal{P}(o'\{x \setminus u\})$.*
(2) *Then, $\mathcal{P}(o\{x \setminus u\}) \simeq \mathcal{P}(o\{x \setminus u'\})$.*

Proof.
(1) By induction on $o \simeq o'$ which, by definition, implies verifying the cases for reflexivity, transitivity, symmetry and congruence (*i.e.* closure by contexts). All are straightforward but the latter.

For congruence we proceed by induction on the closure context $\mathtt{O}$ such that $o = \mathtt{O}\langle p\rangle$ and $o' = \mathtt{O}\langle p'\rangle$ with $p \simeq_* p'$, where $\simeq_*$ is any rule from Figure 5.
- $\mathtt{O} = \Box$. Then, $o = p \simeq_* p' = o'$. Moreover, $o, o' \in \mathbb{T}_{\Lambda M}$. Only two rules are applicable to terms:
  – $\simeq_{\mathtt{exsubs}}$. Follows from Lemma 5.6 (1).
  – $\simeq_{\theta}$. Straightfoward.
- $\mathtt{O} = \boxdot$. Then, $o = p \simeq_* p' = o'$. Moreover, we have $o, o' \in \mathbb{C}_{\Lambda M}$. Three rules are applicable to commands:
  – $\simeq_{\mathtt{exrepl}}$ and $\simeq_{\mathtt{exren}}$. Follows from Lemma 5.6 (2).
  – $\simeq_{\mathtt{ppop}}$. Straightforward.
- $\mathtt{O} = \mathtt{T}\,v$ and $\mathtt{O} = t\,\mathtt{T}$. Both follow from the *i.h.* and Lemma 5.7.
- All the remaining cases follow from the *i.h.*

(2) By induction on o. All cases are by the *i.h.* except for $o = t\,v$ which resorts to Lemma 5.7. □

Lemma D.4. *Let $o, o', s, s' \in \mathbb{O}_{\Lambda M}$ with s and s' stacks such that $o \simeq o'$ and $s \simeq s'$.*
(1) *Then, $\mathcal{P}(o\{\!\{\alpha \backslash^{\alpha'} s\}\!\}) \simeq \mathcal{P}(o'\{\!\{\alpha \backslash^{\alpha'} s\}\!\})$.*
(2) *Then, $\mathcal{P}(o\{\!\{\alpha \backslash^{\alpha'} s\}\!\}) \simeq \mathcal{P}(o\{\!\{\alpha \backslash^{\alpha'} s'\}\!\})$.*

Proof.

(1) By induction on $o \simeq o'$. By definition this implies verifying four conditions: reflexivity, transitivity, symmetry and congruence (*i.e.* closure under contexts). All but the latter are immediate; we thus focus on the latter.

Congruence is proved by induction on the closure context $\mathtt{O}$ such that $o = \mathtt{O}\langle p\rangle$ and $o' = \mathtt{O}\langle p'\rangle$ with $p \simeq_* p'$, where $\simeq_*$ is any rule from Figure 5.

- $\mathtt{O} = \Box$. Follows from Lemma 5.6 (1).
- $\mathtt{O} = \boxdot$. Then, $o = p \simeq_* p' = o'$. Moreover, $o, o' \in \mathbb{C}_{\Lambda M}$. Five rules are applicable to commands:
  - $\simeq_{\mathtt{exrepl}}$. Follow from Lemma 5.6 (2).
  - $\simeq_{\mathtt{exren}}$. Then, $o = [\delta]\,\mu\gamma.\mathtt{LCC}\langle c\rangle$ and $o' = \mathtt{LCC}\langle[\delta]\,\mu\gamma.c\rangle$, and o and o' are in plain form. Without loss of generality, we assume. There are two possible subcases:
    (a) $\delta = \alpha$. Then, $o\{\!\{\alpha \backslash^{\alpha'} s\}\!\} = [\alpha']\,(\mu\gamma.\mathtt{LCC}'\langle c'\rangle) :: s$, where $\mathtt{LCC}' = \mathtt{LCC}\{\!\{\alpha \backslash^{\alpha'} s\}\!\}$, $c' = c\{\!\{\alpha \backslash^{\alpha'} s\}\!\}$ and $o'\{\!\{\alpha \backslash^{\alpha'} s\}\!\} = \mathtt{LCC}\langle[\delta]\,\mu\gamma.c\rangle\{\!\{\alpha \backslash^{\alpha'} s\}\!\}$.

$$
\begin{array}{rcll}
\mathcal{P}(o\{\!\{\alpha \backslash^{\alpha'} s\}\!\}) & = & \mathcal{P}([\alpha']\,(\mu\gamma.\mathtt{LCC}\langle c\rangle\{\!\{\alpha \backslash^{\alpha'} s\}\!\}) :: s) & \\
& = & \mathcal{P}([\alpha']\,\mu\gamma_n.\mathtt{LCC}\langle c\rangle\{\!\{\alpha \backslash^{\alpha'} s\}\!\}[\![\gamma \backslash^{\gamma_1} u_1]\!]\ldots[\![\gamma_{n-1} \backslash^{\gamma_n} u_n]\!]) & \\
& = & \mathcal{P}([\alpha']\,\mu\gamma_n.\mathtt{LCC}'\langle c'\rangle[\![\gamma \backslash^{\gamma_1} u_1]\!]\ldots[\![\gamma_{n-1} \backslash^{\gamma_n} u_n]\!]) & \\
& \simeq & \mathcal{P}(\mathtt{LCC}'\langle[\alpha']\,\mu\gamma_n.c'[\![\gamma \backslash^{\gamma_1} u_1]\!]\ldots[\![\gamma_{n-1} \backslash^{\gamma_n} u_n]\!]\rangle) & \text{(Lemma 5.6)} \\
& = & \mathcal{P}(\mathtt{LCC}'\langle[\alpha']\,(\mu\gamma.c') :: s\rangle) & \\
& = & \mathcal{P}((\mathtt{LCC}\langle[\delta]\,\mu\gamma.c\rangle)\{\!\{\alpha \backslash^{\alpha'} s\}\!\}) & \\
& = & \mathcal{P}(o'\{\!\{\alpha \backslash^{\alpha'} s\}\!\}) &
\end{array}
$$

    (b) $\delta \neq \alpha$. Then, $o\{\!\{\alpha \backslash^{\alpha'} s\}\!\} = [\delta]\,\mu\gamma.\mathtt{LCC}'\langle c'\rangle$, where $\mathtt{LCC}' = \mathtt{LCC}\{\!\{\alpha \backslash^{\alpha'} s\}\!\}$, $c' = c\{\!\{\alpha \backslash^{\alpha'} s\}\!\}$ and $o'\{\!\{\alpha \backslash^{\alpha'} s\}\!\} = \mathtt{LCC}'\langle[\delta]\,\mu\gamma.c'\rangle$.

$$
\begin{array}{rcll}
\mathcal{P}(o\{\!\{\alpha \backslash^{\alpha'} s\}\!\}) & = & \mathcal{P}([\delta]\,\mu\gamma.\mathtt{LCC}'\langle c'\rangle) & \\
& \simeq & \mathcal{P}(\mathtt{LCC}'\langle[\delta]\,\mu\gamma.c'\rangle) & \text{(Lemma 5.6)} \\
& = & \mathcal{P}(o'\{\!\{\alpha \backslash^{\alpha'} s\}\!\}) &
\end{array}
$$

  - $\simeq_{\mathtt{ppop}}$. Then, $o = [\gamma']\,\lambda x.\mu\gamma.[\delta']\,\lambda y.\mu\delta.c$ and $o' = [\delta']\,\lambda y.\mu\delta.[\gamma']\,\lambda x.\mu\gamma.c$ with $\delta \neq \gamma'$ and $\gamma \neq \delta'$. Without loss of generality, we also assume $x \notin s$, $y \notin s$, $\gamma \neq \alpha'$, $\delta \neq \alpha'$, $\gamma \notin s$ and $\delta \notin s$. There are four possible subcases:
    (a) $\gamma' = \alpha$ and $\delta' = \alpha$. Then, $o\{\!\{\alpha \backslash^{\alpha'} s\}\!\} = [\alpha']\,(\lambda x.\mu\gamma.[\alpha']\,(\lambda y.\mu\delta.c') :: s) :: s$ and $o'\{\!\{\alpha \backslash^{\alpha'} s\}\!\} = [\alpha']\,(\lambda y.\mu\delta.[\alpha']\,(\lambda x.\mu\gamma.c') :: s) :: s$ with $c' = c\{\!\{\alpha \backslash^{\alpha'} s\}\!\}$. Suppose $s = u \cdot s'$ (the case $s = u$ is slightly simpler) and consider fresh names γ'', δ''. Then,
    $\mathcal{P}(o\{\!\{\alpha \backslash^{\alpha'} s\}\!\}) = \mathcal{P}([\alpha']\,(\mu\gamma''.([\alpha']\,(\mu\delta''.c'[\![\delta \backslash^{\delta''} s']\!])[y \backslash u])[\![\gamma \backslash^{\gamma''} s']\!])[x \backslash u])$
    where the explicit substitutions result from contracting the $\mathtt{dB}$-redexes in u and the explicit replacements result from contracting the $\mathtt{dM}$-redexes in s' followed by the resulting $\mathtt{C}$-redexes. If $s = u$, then there are no $\mathtt{dM}$-redexes to contract, and there are only explicit substitutions. Similarly,

$\mathcal{P}(o'\{\!\{\alpha \setminus^{\alpha'} s\}\!\}) = \mathcal{P}([\alpha']\,(\mu\delta''.([\alpha']\,(\mu\gamma''.c'[\![\gamma \setminus^{\gamma''} s']\!])[x \setminus u])[\![\delta \setminus^{\delta''} s']\!])[y \setminus u])$
Let $c'' = c'[\![\gamma \setminus^{\gamma''} s']\!][\![\delta \setminus^{\delta''} s']\!]$. Then by Lemma 5.6, on the one hand we have: $\mathcal{P}(o\{\!\{\alpha \setminus^{\alpha'} s\}\!\}) \simeq \mathcal{P}([\alpha']\,\mu\gamma''.[\alpha']\,(\mu\delta''.c'')[x \setminus u][y \setminus u])$, and on the other hand we have: $\mathcal{P}(o'\{\!\{\alpha \setminus^{\alpha'} s\}\!\}) \simeq \mathcal{P}([\alpha']\,(\mu\delta''.[\alpha']\,\mu\gamma''.c'')[x \setminus u][y \setminus u])$; resorting to item (1) of that lemma to correctly place the explicit substitutions and item (2). We then conclude by applying $\simeq_{\mathtt{exren}}$:

$$\begin{array}{rcl}\mathcal{P}(o\{\!\{\alpha \setminus^{\alpha'} s\}\!\}) & \simeq & \mathcal{P}([\alpha']\,\mu\gamma''.[\alpha']\,(\mu\delta''.c'')[x \setminus u][y \setminus u]) \\ & = & [\alpha']\,\mu\gamma''.[\alpha']\,(\mu\delta''.\mathcal{P}(c''))[x \setminus \mathcal{P}(u)][y \setminus \mathcal{P}(u)] \\ & \simeq_{\mathtt{exren}} & [\alpha']\,(\mu\delta''.[\alpha']\,\mu\gamma''.\mathcal{P}(c''))[x \setminus \mathcal{P}(u)][y \setminus \mathcal{P}(u)] \\ & = & \mathcal{P}([\alpha']\,(\mu\delta''.[\alpha']\,\mu\gamma''.c'')[x \setminus u][y \setminus u]) \\ & \simeq & \mathcal{P}(o'\{\!\{\alpha \setminus^{\alpha'} s\}\!\})\end{array}$$

(b) $\gamma' = \alpha$ and $\delta' \neq \alpha$, and the case $\gamma' \neq \alpha$ and $\delta' = \alpha$. Similar to the previous case.
(c) $\gamma' \neq \alpha$ and $\delta' \neq \alpha$. Straightforward.

- $\mathtt{O} = \mathtt{T}\,u$ and $\mathtt{O} = t\,\mathtt{T}$. Follow from Lemma 5.7.
- $\mathtt{O} = \lambda x.\mathtt{T}$, $\mathtt{O} = \mu\gamma.\mathtt{C}$, $\mathtt{O} = \mathtt{T}[x \setminus u]$, and $\mathtt{O} = t[x \setminus \mathtt{T}]$. Similar to the previous case.
- $\mathtt{O} = [\gamma]\,\mathtt{T}$. If $\gamma \neq \alpha$, then we conclude from the *i.h.* ; otherwise we reason as follows. First note that $\mathcal{P}(\mathtt{T}\langle p\rangle\{\!\{\alpha \setminus^{\alpha'} s\}\!\}) \simeq_{i.h.} \mathcal{P}(\mathtt{T}\langle p'\rangle\{\!\{\alpha \setminus^{\alpha'} s\}\!\})$. Then $\mathcal{P}(o\{\!\{\alpha \setminus^{\alpha'} s\}\!\}) = \mathcal{P}([\alpha']\,\mathcal{P}((\mathtt{T}\langle p\rangle\{\!\{\alpha \setminus^{\alpha'} s\}\!\})) :: s) \simeq \mathcal{P}([\alpha']\,\mathcal{P}((\mathtt{T}\langle p'\rangle\{\!\{\alpha \setminus^{\alpha'} s\}\!\})) :: s) = \mathcal{P}(o'\{\!\{\alpha \setminus^{\alpha'} s\}\!\})$ by Lemma 5.7.
- $\mathtt{O} = \mathtt{C}[\![\gamma \setminus^{\delta} s']\!]$ and $\mathtt{O} = c[\![\gamma \setminus^{\delta} \mathtt{S}]\!]$. By the *i.h.*
- $\mathtt{O} = \mathtt{T} \cdot s'$ and $\mathtt{O} = t \cdot \mathtt{S}$. We use the *i.h.*

(2) By induction on o. All cases follow from the *i.h.* and/or Lemma 5.7. □

Lemma 5.8. *Let $u, s, o \in \mathbb{O}_{\Lambda M}$ in $\mathcal{P}$-normal form such that $o \simeq o'$, $u \simeq u'$ and $s \simeq s'$, where u, u' are terms and s, s' are stacks. Then,*
(1) $\mathcal{P}(o\{x \setminus u\}) \simeq \mathcal{P}(o'\{x \setminus u\})$ *and* $\mathcal{P}(o\{x \setminus u\}) \simeq \mathcal{P}(o\{x \setminus u'\})$.
(2) $\mathcal{P}(o\{\!\{\alpha \setminus^{\alpha'} s\}\!\}) \simeq \mathcal{P}(o'\{\!\{\alpha \setminus^{\alpha'} s\}\!\})$ *and* $\mathcal{P}(o\{\!\{\alpha \setminus^{\alpha'} s\}\!\}) \simeq \mathcal{P}(o\{\!\{\alpha \setminus^{\alpha'} s'\}\!\})$.

Proof. By Lemma D.3 and D.4, resp. □

Theorem 5.9. *Let $o, p \in \mathbb{O}_{\Lambda M}$. If $o \simeq p$ and $o \rightsquigarrow o'$, then there exists p' such that $p \rightsquigarrow p'$ and $o' \simeq p'$.*

Proof. By induction on $o \simeq p$. By definition this requires verifying four cases: reflexivity, transitivity, symmetry and congruence (*i.e.* closure under contexts). The cases where $o \simeq p$ holds by reflexivity or transitivity are straightforward. We focus on the other two, which are dealt with simultaneously by induction on the closure context $\mathtt{Q}$ such that $o = \mathtt{Q}\langle q\rangle$ and $p = \mathtt{Q}\langle q'\rangle$ with $q \simeq_* q'$, where $\simeq_*$ is any rule in Figure 5. Note that $o \rightsquigarrow o'$ implies $o = \mathtt{O}\langle l\rangle$ and $o' = \mathcal{P}(\mathtt{O}\langle r\rangle)$ with $l \mapsto_* r$, $* \in \{\mathtt{S}, \mathtt{R}^\bullet\}$. We thus consider all possible forms for $\mathtt{Q}$ and $\mathtt{O}$:

- $\mathtt{Q} = \Box$. Then $o = q \simeq_* q' = p$. Moreover, in this case $o, p \in \mathbb{T}_{\Lambda M}$. We only detail the cases where there is an overlap between the equivalence and the reduction rules, the others being immediate. Only two rules are applicable to terms:
  - $\simeq_{\mathtt{exsubs}}$. Then, $o = \mathtt{LTT}\langle t\rangle[x \setminus u]$ and $p = \mathtt{LTT}\langle t[x \setminus u]\rangle$ with $x \notin \mathtt{LTT}$ and $\mathtt{fc}(u, \mathtt{LTT})$. There are three further possible cases:

(1) $\mathtt{S}$-redex at the root.

$$\begin{array}{ccc} o = \mathtt{LTT}\langle t\rangle[x \setminus u] & \simeq_{\mathtt{exsubs}} & \mathtt{LTT}\langle t[x \setminus u]\rangle = p \\ \mathtt{S}\downarrow & & \mathtt{S}\downarrow \\ o' = \mathcal{P}(\mathtt{LTT}\langle t\rangle\{x \setminus u\}) & = & \mathcal{P}(\mathtt{LTT}\langle t\{x \setminus u\}\rangle) = p' \end{array}$$

(2) $\mathtt{S}$-redex overlaps $\mathtt{LTT}$. Then we have the context $\mathtt{LTT} = \mathtt{LTT}_1\langle \mathtt{LTT}_2[y \setminus v]\rangle$ with commands $o' = \mathcal{P}(\mathtt{LTT}_1\langle \mathtt{LTT}_2\langle t\rangle\{y \setminus v\}\rangle[x \setminus u]) = \mathcal{P}(\mathtt{LTT}_1\langle \mathtt{LTT}'_2\langle t'\rangle\rangle[x \setminus u])$ and $p' = \mathcal{P}(\mathtt{LTT}_1\langle \mathtt{LTT}'_2\langle t'[x \setminus u]\rangle\rangle)$ since $\mathsf{fc}(u, \mathtt{LTT})$ implies $y \notin u$. We conclude by Lemma 5.6 (1):

$$\begin{array}{ccc} o = \mathtt{LTT}_1\langle \mathtt{LTT}_2\langle t\rangle[y \setminus v]\rangle[x \setminus u] & \simeq_{\mathtt{exsubs}} & \mathtt{LTT}_1\langle \mathtt{LTT}_2\langle t[x \setminus u]\rangle[y \setminus v]\rangle = p \\ \mathtt{S}\downarrow & & \mathtt{S}\downarrow \\ o' = \mathcal{P}(\mathtt{LTT}_1\langle \mathtt{LTT}_2\langle t\rangle\{y \setminus v\}\rangle[x \setminus u]) & \simeq & \mathcal{P}(\mathtt{LTT}_1\langle \mathtt{LTT}_2\langle t[x \setminus u]\rangle\{y \setminus v\}\rangle) = p' \end{array}$$

(3) $\mathtt{R}^\bullet$-redex overlaps $\mathtt{LTT}$. Then we have the context $\mathtt{LTT} = \mathtt{LTC}\langle \mathtt{LCT}[\![\alpha \setminus^{\alpha'} s]\!]\rangle$ with commands $o' = \mathcal{P}(\mathtt{LTC}\langle \mathtt{LCT}\langle t\rangle\{\!\{\alpha \setminus^{\alpha'} s\}\!\}\rangle[x \setminus u]) = \mathcal{P}(\mathtt{LTC}\langle \mathtt{LTC}'\langle t'\rangle\rangle[x \setminus u])$ and $p' = \mathcal{P}(\mathtt{LTC}\langle \mathtt{LTC}'\langle t'[x \setminus u]\rangle\rangle)$ since $\mathsf{fc}(u, \mathtt{LTT})$ implies $\alpha \notin u$. We conclude by Lemma 5.6 (1):

$$\begin{array}{ccc} o = \mathtt{LTC}\langle \mathtt{LCT}\langle t\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle[x \setminus u] & \simeq_{\mathtt{exsubs}} & \mathtt{LTC}\langle \mathtt{LCT}\langle t[x \setminus u]\rangle[\![\alpha \setminus^{\alpha'} s]\!]\rangle = p \\ \mathtt{R}^\bullet\downarrow & & \mathtt{R}^\bullet\downarrow \\ o' = \mathcal{P}(\mathtt{LTC}\langle \mathtt{LCT}\langle t\rangle\{\!\{\alpha \setminus^{\alpha'} s\}\!\}\rangle[x \setminus u]) & \simeq & \mathcal{P}(\mathtt{LTC}\langle \mathtt{LCT}\langle t[x \setminus u]\rangle\{\!\{\alpha \setminus^{\alpha'} s\}\!\}\rangle) = p' \end{array}$$

– $\simeq_\theta$. Then, $o = \mu\alpha.[\alpha]\, t$ and $p = t$ with $\alpha \notin t$. This case is immediate since all reduction steps must be in t.

- $\mathtt{Q} = \boxdot$. Then, $o = q \simeq_* q' = p$. Moreover, in this case $o, p \in \mathbb{C}_{\Lambda M}$. We only detail the cases where there is an overlap between the equivalence and the reduction rules, the others being immediate. There are three rules applicable to commands:

– $\simeq_{\mathtt{exrepl}}$. Then, $o = \mathtt{LCC}\langle c\rangle[\![\alpha \setminus^{\alpha'} s]\!]$ and $p = \mathtt{LCC}\langle c[\![\alpha \setminus^{\alpha'} s]\!]\rangle$ with $\alpha \notin \mathtt{LCC}$, $\mathsf{fc}(\alpha', \mathtt{LCC})$ and $\mathsf{fc}(s, \mathtt{LCC})$. There are three further possible cases:

(1) $\mathtt{R}^\bullet$-redex at the root.

$$\begin{array}{ccc} o = \mathtt{LCC}\langle c\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \simeq_{\mathtt{exrepl}} & \mathtt{LCC}\langle c[\![\alpha \setminus^{\alpha'} s]\!]\rangle = p \\ \mathtt{R}^\bullet\downarrow & & \mathtt{R}^\bullet\downarrow \\ o' = \mathcal{P}(\mathtt{LCC}\langle c\rangle\{\!\{\alpha \setminus^{\alpha'} s\}\!\}) & = & \mathcal{P}(\mathtt{LCC}\langle c\{\!\{\alpha \setminus^{\alpha'} s\}\!\}\rangle) = p' \end{array}$$

(2) $\mathtt{S}$-redex overlaps $\mathtt{LCC}$. Then we have the context $\mathtt{LCC} = \mathtt{LCT}\langle \mathtt{LTC}[x \setminus u]\rangle$ with commands $o' = \mathcal{P}(\mathtt{LCT}\langle \mathtt{LTC}\langle c\rangle\{x \setminus u\}\rangle[\![\alpha \setminus^{\alpha'} s]\!]) = \mathcal{P}(\mathtt{LCT}\langle \mathtt{LTC}'\langle c'\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!])$ and $p' = \mathcal{P}(\mathtt{LCT}\langle \mathtt{LTC}'\langle c'[\![\alpha \setminus^{\alpha'} s]\!]\rangle\rangle)$ since $\mathsf{fc}(s, \mathtt{LCC})$ implies $x \notin s$. We conclude by Lemma 5.6 (2):

$$\begin{array}{ccc} o = \mathtt{LCT}\langle \mathtt{LTC}\langle c\rangle[x \setminus u]\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \simeq_{\mathtt{exrepl}} & \mathtt{LCT}\langle \mathtt{LTC}\langle c[\![\alpha \setminus^{\alpha'} s]\!]\rangle[x \setminus u]\rangle = p \\ \mathtt{S}\downarrow & & \mathtt{S}\downarrow \\ o' = \mathcal{P}(\mathtt{LCT}\langle \mathtt{LTC}\langle c\rangle\{x \setminus u\}\rangle[\![\alpha \setminus^{\alpha'} s]\!]) & \simeq & \mathcal{P}(\mathtt{LCT}\langle \mathtt{LTC}\langle c[\![\alpha \setminus^{\alpha'} s]\!]\rangle\{x \setminus u\}\rangle) = p' \end{array}$$

(3) $\mathtt{R}^\bullet$-redex overlaps $\mathtt{LCC}$. Then we have the context $\mathtt{LCC} = \mathtt{LCC}_1\langle \mathtt{LCC}_2[\![\gamma \setminus^\delta s']\!]\rangle$ with commands $o' = \mathcal{P}(\mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c\rangle\{\!\{\gamma \setminus^\delta s'\}\!\}\rangle[\![\alpha \setminus^{\alpha'} s]\!]) = \mathcal{P}(\mathtt{LCC}_1\langle \mathtt{LCC}'_2\langle c'\rangle\rangle[\![\alpha \setminus^{\alpha'} s]\!])$ and $p' = \mathcal{P}(\mathtt{LCC}_1\langle \mathtt{LCC}'_2\langle c'[\![\alpha \setminus^{\alpha'} s]\!]\rangle\rangle)$ since $\mathsf{fc}(\alpha, \mathtt{LCC})$ implies $\gamma \neq \alpha$, and $\mathsf{fc}(s, \mathtt{LCC})$ implies $\alpha \notin s$. We conclude by Lemma 5.6 (2):

$$\begin{array}{ccc}
o = \mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c\rangle[\![\gamma \setminus^\delta s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!] & \simeq_{\mathtt{exrepl}} & \mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c[\![\alpha \setminus^{\alpha'} s]\!]\rangle[\![\gamma \setminus^\delta s']\!]\rangle = p \\
\mathtt{R}^\bullet \rightsquigarrow & & \mathtt{R}^\bullet \rightsquigarrow \\
o' = \mathcal{P}(\mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c\rangle\{\!\{\gamma \setminus^\delta s'\}\!\}\rangle[\![\alpha \setminus^{\alpha'} s]\!]) & \simeq & \mathcal{P}(\mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c[\![\alpha \setminus^{\alpha'} s]\!]\rangle\{\!\{\gamma \setminus^\delta s'\}\!\}\rangle) = p'
\end{array}$$

– $\simeq_{\mathtt{exren}}$. Then, $o = [\beta]\,\mu\alpha.\mathtt{LCC}\langle c\rangle$ and $p = \mathtt{LCC}\langle[\beta]\,\mu\alpha.c\rangle$ with $\alpha \notin \mathtt{LCC}, \mathsf{fc}(\beta, \mathtt{LCC})$. There are two further possible cases:

(1) $\mathtt{S}$-redex overlaps $\mathtt{LCC}$. Then we have the context $\mathtt{LCC} = \mathtt{LCT}\langle \mathtt{LTC}[x \setminus u]\rangle$ with commands $o' = \mathcal{P}([\beta]\,\mu\alpha.\mathtt{LCT}\langle \mathtt{LTC}\langle c\rangle\{x \setminus u\}\rangle) = \mathcal{P}([\beta]\,\mu\alpha.\mathtt{LCT}\langle \mathtt{LTC}'\langle c'\rangle\rangle)$ and $p' = \mathcal{P}(\mathtt{LCT}\langle \mathtt{LTC}'\langle[\beta]\,\mu\alpha.c'\rangle\rangle)$. We conclude by Lemma 5.6 (2):

$$\begin{array}{ccc}
o = [\beta]\,\mu\alpha.\mathtt{LCT}\langle \mathtt{LTC}\langle c\rangle[x \setminus u]\rangle & \simeq_{\mathtt{exren}} & \mathtt{LCT}\langle \mathtt{LTC}\langle[\beta]\,\mu\alpha.c\rangle[x \setminus u]\rangle = p \\
\mathtt{S} \rightsquigarrow & & \mathtt{S} \rightsquigarrow \\
o' = \mathcal{P}([\beta]\,\mu\alpha.\mathtt{LCT}\langle \mathtt{LTC}\langle c\rangle\{x \setminus u\}\rangle) & \simeq & \mathcal{P}(\mathtt{LCT}\langle \mathtt{LTC}\langle[\beta]\,\mu\alpha.c\rangle\{x \setminus u\}\rangle) = p'
\end{array}$$

(2) $\mathtt{R}^\bullet$-redex overlaps $\mathtt{LCC}$. Then we have the context $\mathtt{LCC} = \mathtt{LCC}_1\langle \mathtt{LCC}_2[\![\gamma \setminus^\delta s]\!]\rangle$ with commands $o' = \mathcal{P}([\beta]\,\mu\alpha.\mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c\rangle\{\!\{\gamma \setminus^\delta s\}\!\}\rangle) = \mathcal{P}([\beta]\,\mu\alpha.\mathtt{LCC}_1\langle \mathtt{LCC}'_2\langle c'\rangle\rangle)$ and $p' = \mathcal{P}(\mathtt{LCC}_1\langle \mathtt{LCC}'_2\langle[\beta]\,\mu\alpha.c'\rangle\rangle)$ since $\mathsf{fc}(\beta, \mathtt{LCC})$ implies $\gamma \neq \beta$. We conclude by Lemma 5.6 (2):

$$\begin{array}{ccc}
o = [\beta]\,\mu\alpha.\mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c\rangle[\![\gamma \setminus^\delta s]\!]\rangle & \simeq_{\mathtt{exren}} & \mathtt{LCC}_1\langle \mathtt{LCC}_2\langle[\beta]\,\mu\alpha.c\rangle[\![\gamma \setminus^\delta s]\!]\rangle = p \\
\mathtt{R}^\bullet \rightsquigarrow & & \mathtt{R}^\bullet \rightsquigarrow \\
o' = \mathcal{P}([\beta]\,\mu\alpha.\mathtt{LCC}_1\langle \mathtt{LCC}_2\langle c\rangle\{\!\{\gamma \setminus^\delta s\}\!\}\rangle) & \simeq & \mathcal{P}(\mathtt{LCC}_1\langle \mathtt{LCC}_2\langle[\beta]\,\mu\alpha.c\rangle\{\!\{\gamma \setminus^\delta s\}\!\}\rangle) = p'
\end{array}$$

– $\simeq_{\mathtt{ppop}}$. Then, $o = [\alpha']\,\lambda z.\mu\alpha.[\beta']\,\lambda y.\mu\beta.c$ and $p = [\beta']\,\lambda y.\mu\beta.[\alpha']\,\lambda z.\mu\alpha.c$ with $\beta \neq \alpha'$ and $\alpha \neq \beta'$. This case is immediate since all possible reductions are in c.

- $\mathtt{Q} = \mathtt{T}\,u$ or $\mathtt{Q} = t\,\mathtt{T}$. We use the *i.h.* and Lemma 5.7.
- $\mathtt{Q} = \lambda x.\mathtt{T}$ and $\mathtt{Q} = \mu\alpha.\mathtt{C}$. By the *i.h.*
- $\mathtt{Q} = \mathtt{T}[x \setminus u]$. Then, $o = \mathtt{T}\langle q\rangle[x \setminus u]$ and $p = \mathtt{T}\langle q'\rangle[x \setminus u]$. If the $\rightsquigarrow_{\mathtt{S}}$ step executes the outermost explicit substitution, we use Lemma D.3 (1), otherwise we use Lemma 5.7 and the *i.h.*
- $\mathtt{Q} = t[x \setminus \mathtt{T}]$. Then, $o = t[x \setminus \mathtt{T}\langle q\rangle]$ and $p = t[x \setminus \mathtt{T}\langle q'\rangle]$. If the $\rightsquigarrow_{\mathtt{S}}$ step executes the outermost explicit substitution, we use Lemma D.3 (2), otherwise we use Lemma 5.7 and the *i.h.*
- $\mathtt{Q} = [\alpha]\,\mathtt{T}$. By the *i.h.*
- $\mathtt{Q} = \mathtt{C}[\![\alpha \setminus^{\alpha'} s]\!]$. Then, $o = \mathtt{C}\langle q\rangle[\![\alpha \setminus^{\alpha'} s]\!]$ and $p = \mathtt{C}\langle q'\rangle[\![\alpha \setminus^{\alpha'} s]\!]$. If the $\rightsquigarrow_{\mathtt{R}^\bullet}$ step executes the outermost explicit replacement, we use Lemma D.4 (1), otherwise we use Lemma 5.7 and the *i.h.*
- $\mathtt{Q} = s[\![\alpha \setminus^{\alpha'} \mathtt{S}]\!]$. Then, $o = c[\![\alpha \setminus^{\alpha'} \mathtt{C}\langle q\rangle]\!]$ and $p = c[\![\alpha \setminus^{\alpha'} \mathtt{C}\langle q'\rangle]\!]$. If the $\rightsquigarrow_{\mathtt{R}^\bullet}$ step executes the outermost explicit replacement, we use Lemma D.4 (2), otherwise we use Lemma 5.7 and the *i.h.* □

Appendix E. Correspondence Results

Lemma 6.2. *Let $t, u \in \mathbb{T}_{\lambda\mu}$ and $c \in \mathbb{C}_{\lambda\mu}$. Let s be a stack of terms in $\mathbb{T}_{\lambda\mu}$. Let $x \notin s, \alpha \notin s', \beta \notin s, \beta \neq \alpha', \alpha \neq \beta'$. Then,*

(1) $((\lambda x.t)\,u) :: s \simeq_\tau (\lambda x.t :: s)\,u$
(2) $[\alpha']\,(\mu\alpha.[\beta']\,(\mu\beta.c) :: s') :: s \simeq_\tau [\beta']\,(\mu\beta.[\alpha']\,(\mu\alpha.c) :: s) :: s'$
(3) $[\alpha']\,(\mu\alpha.[\beta']\,\lambda x.\mu\beta.c) :: s \simeq_\tau [\beta']\,\lambda x.\mu\beta.[\alpha']\,(\mu\alpha.c) :: s$
(4) $[\alpha']\,(\mu\alpha.[\beta']\,\mu\beta.c) :: s \simeq_\tau [\beta']\,\mu\beta.[\alpha']\,(\mu\alpha.c) :: s$

Proof.

(1) By induction on s.
- $s = v$. Then,

$$((\lambda x.t)\,u) :: s = ((\lambda x.t)\,u)\,v \simeq_{\tau_2} (\lambda x.t\,v)\,u = (\lambda x.t :: s)\,u$$

- $s = v \cdot s'$. Then,

$$\begin{array}{rcll} ((\lambda x.t)\,u) :: s & = & (((\lambda x.t)\,u)\,v) :: s' & \\ & \simeq_{\tau_2} & ((\lambda x.t\,v)\,u) :: s' & \\ & \simeq_{\tau} & (\lambda x.(t\,v) :: s')\,u & (i.h.) \\ & = & (\lambda x.t :: s)\,u & \end{array}$$

(2) By induction on s.
- $s = v$. By induction once again, this time on s'.
  - $s' = w$. Then,

$$\begin{array}{rcl} [\alpha']\,(\mu\alpha.[\beta']\,(\mu\beta.c) :: s') :: s & = & [\alpha']\,(\mu\alpha.[\beta']\,(\mu\beta.c)\,w)\,v \\ & \simeq_{\tau_4} & [\beta']\,(\mu\beta.[\alpha']\,(\mu\alpha.c)\,v)\,w \\ & = & [\beta']\,(\mu\beta.[\alpha']\,(\mu\alpha.c) :: s) :: s' \end{array}$$

  - $s' = w \cdot s''$. Then,

$$\begin{array}{rcll} [\alpha']\,(\mu\alpha.[\beta']\,(\mu\beta.c) :: s') :: s & = & [\alpha']\,(\mu\alpha.[\beta']\,((\mu\beta.c)\,w) :: s'')\,v & \\ & \simeq_{\tau_7} & [\alpha']\,(\mu\alpha.[\beta']\,(\mu\gamma.[\gamma]\,(\mu\beta.c)\,w) :: s'')\,v & (\gamma \text{ fresh}) \\ & \simeq_{\tau} & [\beta']\,(\mu\gamma.[\alpha']\,(\mu\alpha.[\gamma]\,(\mu\beta.c)\,w)\,v) :: s'' & (i.h.) \\ & \simeq_{\tau_4} & [\beta']\,(\mu\gamma.[\gamma]\,(\mu\beta.[\alpha']\,(\mu\alpha.c)\,v)\,w) :: s'' & \\ & \simeq_{\tau_7} & [\beta']\,((\mu\beta.[\alpha']\,(\mu\alpha.c)\,v)\,w) :: s'' & \\ & = & [\beta']\,(\mu\beta.[\alpha']\,(\mu\alpha.c) :: s) :: s' & \end{array}$$

- $s = v \cdot s''$. Then,

$$\begin{array}{rcll} [\alpha']\,(\mu\alpha.[\beta']\,(\mu\beta.c) :: s') :: s & = & [\alpha']\,((\mu\alpha.[\beta']\,(\mu\beta.c) :: s')\,v) :: s'' & \\ & \simeq_{\tau_7} & [\alpha']\,(\mu\gamma.[\gamma]\,(\mu\alpha.[\beta']\,(\mu\beta.c) :: s')\,v) :: s'' & (\gamma \text{ fresh}) \\ & \simeq_{\tau} & [\alpha']\,(\mu\gamma.[\beta']\,(\mu\beta.[\gamma]\,(\mu\alpha.c)\,v) :: s') :: s'' & (i.h.) \\ & \simeq_{\tau} & [\beta']\,(\mu\beta.[\alpha']\,(\mu\gamma.[\gamma]\,(\mu\alpha.c)\,v) :: s'') :: s' & (i.h.) \\ & \simeq_{\tau_7} & [\beta']\,(\mu\beta.[\alpha']\,((\mu\alpha.c)\,v) :: s'') :: s' & \\ & = & [\beta']\,(\mu\beta.[\alpha']\,(\mu\alpha.c) :: s) :: s' & \end{array}$$

(3) By induction on s.
- $s = v$. Then,

$$\begin{array}{rcl} [\alpha']\,(\mu\alpha.[\beta']\,\lambda x.\mu\beta.c) :: s & = & [\alpha']\,(\mu\alpha.[\beta']\,\lambda x.\mu\beta.c)\,v \\ & \simeq_{\tau_5} & [\beta']\,\lambda x.\mu\beta.[\alpha']\,(\mu\alpha.c)\,v \\ & = & [\beta']\,\lambda x.\mu\beta.[\alpha']\,(\mu\alpha.c) :: s \end{array}$$

- $s = v \cdot s'$. Then,

$$\begin{array}{rcll} [\alpha'](\mu\alpha.[\beta']\lambda x.\mu\beta.c) :: s & = & [\alpha']((\mu\alpha.[\beta']\lambda x.\mu\beta.c)\,v) :: s' & \\ & \simeq_{\tau_7} & [\alpha'](\mu\gamma.[\gamma](\mu\alpha.[\beta']\lambda x.\mu\beta.c)\,v) :: s' & (\gamma \text{ fresh}) \\ & \simeq_{\tau_5} & [\alpha'](\mu\gamma.[\beta']\lambda x.\mu\beta.[\gamma](\mu\alpha.c)\,v) :: s' & \\ & \simeq_{\tau} & [\beta']\lambda x.\mu\beta.[\alpha'](\mu\gamma.[\gamma](\mu\alpha.c)\,v) :: s' & (\textit{i.h.}) \\ & \simeq_{\tau_7} & [\beta']\lambda x.\mu\beta.[\alpha']((\mu\alpha.c)\,v) :: s' & \\ & = & [\beta']\lambda x.\mu\beta.[\alpha'](\mu\alpha.c) :: s & \end{array}$$

(4) By induction on s.
- $s = v$. Then,

$$\begin{array}{rcl} [\alpha'](\mu\alpha.[\beta']\mu\beta.c) :: s & = & [\alpha'](\mu\alpha.[\beta']\mu\beta.c)\,v \\ & \simeq_{\tau_{10}} & [\beta']\mu\beta.[\alpha'](\mu\alpha.c)\,v \\ & = & [\beta']\mu\beta.[\alpha'](\mu\alpha.c) :: s \end{array}$$

- $s = v \cdot s'$. Then,

$$\begin{array}{rcll} [\alpha'](\mu\alpha.[\beta']\mu\beta.c) :: s & = & [\alpha']((\mu\alpha.[\beta']\mu\beta.c)\,v) :: s' & \\ & \simeq_{\tau_7} & [\alpha'](\mu\gamma.[\gamma](\mu\alpha.[\beta']\mu\beta.c)\,v) :: s' & (\gamma \text{ fresh}) \\ & \simeq_{\tau_{10}} & [\alpha'](\mu\gamma.[\beta']\mu\beta.[\gamma](\mu\alpha.c)\,v) :: s' & \\ & \simeq_{\tau} & [\beta']\mu\beta.[\alpha'](\mu\gamma.[\gamma](\mu\alpha.c)\,v) :: s' & (\textit{i.h.}) \\ & \simeq_{\tau_7} & [\beta']\mu\beta.[\alpha']((\mu\alpha.c)\,v) :: s' & \\ & = & [\beta']\mu\beta.[\alpha'](\mu\alpha.c) :: s & \end{array}$$

□

Lemma 6.4. *Let $t \in \mathbb{T}_{\lambda\mu}$ and $c \in \mathbb{C}_{\lambda\mu}$. Let s, s' be stacks and* LCC *be a **CC** linear context. Then,*

(1) $\mathtt{e}(\mathtt{LCC}\langle[\alpha]\,t\rangle[\![\alpha \setminus^{\alpha'} s]\!]) \simeq_\tau \mathtt{e}(\mathtt{LCC}\langle[\alpha']\,t :: s\rangle)$, *where* $\alpha \notin \mathtt{LCC}$ *and* $\alpha \notin t$.
(2) $\mathtt{e}(\mathtt{LCC}\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!]) \simeq_\tau \mathtt{e}(\mathtt{LCC}\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle) = o'$, *where* $\alpha \notin \mathtt{LCC}$, $\alpha \notin c$ *and* $\alpha \notin s'$.

Proof.

(1) We address item (1) first, by induction on the size of LCC.
- $\mathtt{LCC} = \boxdot$.

$$\begin{array}{rcl} \mathtt{e}(([\alpha]\,t)[\![\alpha \setminus^{\alpha'} s]\!]) & = & [\alpha']\,\mu\alpha.[\alpha]\,(\mathtt{e}(t) :: \mathtt{e}(s)) \\ & \simeq_{\tau_7} & [\alpha']\,\mathtt{e}(t) :: \mathtt{e}(s) \\ & = & \mathtt{e}([\alpha']\,t :: s) \end{array}$$

- $\mathtt{LCC} = [\beta]\,\mathtt{LTC}$. We proceed by analyzing the shape of LTC.
  - $\mathtt{LTC} = \mu\gamma.\mathtt{LCC}'$.

$$\begin{array}{rll} & \mathtt{e}(([\beta]\,\mu\gamma.\mathtt{LCC}'\langle[\alpha]\,t\rangle)[\![\alpha \setminus^{\alpha'} s]\!]) & \\ = & [\alpha'](\mu\alpha.[\beta]\,\mu\gamma.\mathtt{e}(\mathtt{LCC}'\langle[\alpha]\,t\rangle)) :: \mathtt{e}(s) & \\ \simeq_\tau & [\beta]\,\mu\gamma.[\alpha'](\mu\alpha.\mathtt{e}(\mathtt{LCC}'\langle[\alpha]\,t\rangle)) :: \mathtt{e}(s) & (\text{Lemma 6.2:4}) \\ = & [\beta]\,\mu\gamma.\mathtt{e}(\mathtt{LCC}'\langle[\alpha]\,t\rangle[\![\alpha \setminus^{\alpha'} s]\!]) & \\ \simeq_\tau & [\beta]\,\mu\gamma.\mathtt{e}(\mathtt{LCC}'\langle[\alpha']\,t :: s\rangle) & (\textit{i.h.}) \\ = & \mathtt{e}([\beta]\,\mu\gamma.\mathtt{LCC}'\langle[\alpha']\,t :: s\rangle) & \\ = & \mathtt{e}(\mathtt{LCC}\langle[\alpha']\,t :: s\rangle) & \end{array}$$

- $\mathtt{LTC} = \mathtt{LTC}'\, w$.

$$
\begin{array}{lll}
 & \mathsf{e}(([\beta]\,\mathtt{LTC}'\langle[\alpha]\,t\rangle\,w)[\![\alpha \setminus^{\alpha'} s]\!]) & \\
= & [\alpha']\,(\mu\alpha.[\beta]\,\mathsf{e}(\mathtt{LTC}'\langle[\alpha]\,t\rangle)\,\mathsf{e}(w)) :: \mathsf{e}(s) & \\
\simeq_{\tau_7} & [\alpha']\,(\mu\alpha.[\beta]\,(\mu\gamma.[\gamma]\,\mathsf{e}(\mathtt{LTC}'\langle[\alpha]\,t\rangle))\,\mathsf{e}(w)) :: \mathsf{e}(s) & (\gamma \text{ fresh}) \\
\simeq_{\tau} & [\beta]\,(\mu\gamma.[\alpha']\,(\mu\alpha.[\gamma]\,\mathsf{e}(\mathtt{LTC}'\langle[\alpha]\,t\rangle)) :: \mathsf{e}(s))\,\mathsf{e}(w) & (\text{Lemma 6.2:2}) \\
= & [\beta]\,(\mu\gamma.\mathsf{e}(([\gamma]\,\mathtt{LTC}'\langle[\alpha]\,t\rangle)[\![\alpha \setminus^{\alpha'} s]\!]))\,\mathsf{e}(w) & \\
\simeq_{\tau} & [\beta]\,(\mu\gamma.\mathsf{e}([\gamma]\,\mathtt{LTC}'\langle[\alpha']\,t :: s\rangle))\,\mathsf{e}(w) & (\textit{i.h.}) \\
= & [\beta]\,(\mu\gamma.[\gamma]\,\mathsf{e}(\mathtt{LTC}'\langle[\alpha']\,t :: s\rangle))\,\mathsf{e}(w) & \\
\simeq_{\tau_7} & [\beta]\,\mathsf{e}(\mathtt{LTC}'\langle[\alpha']\,t :: s\rangle)\,\mathsf{e}(w) & \\
= & \mathsf{e}([\beta]\,\mathtt{LTC}'\langle[\alpha']\,t :: s\rangle\,w) & \\
= & \mathsf{e}(\mathtt{LCC}\langle[\alpha']\,t :: s\rangle) &
\end{array}
$$

Note that the *i.h.* applies since $\mathsf{sz}([\gamma]\,\mathtt{LTC}') < \mathsf{sz}(\mathtt{LCC})$.

- $\mathtt{LTC} = \lambda x.\mathtt{LTC}'$.

$$
\begin{array}{lll}
 & \mathsf{e}(([\beta]\,\lambda x.\mathtt{LTC}'\langle[\alpha]\,t\rangle)[\![\alpha \setminus^{\alpha'} s]\!]) & \\
= & [\alpha']\,(\mu\alpha.[\beta]\,\lambda x.\mathsf{e}(\mathtt{LTC}'\langle[\alpha]\,t\rangle)) :: \mathsf{e}(s) & \\
\simeq_{\tau_7} & [\alpha']\,(\mu\alpha.[\beta]\,\lambda x.\mu\gamma.[\gamma]\,\mathsf{e}(\mathtt{LTC}'\langle[\alpha]\,t\rangle)) :: \mathsf{e}(s) & (\gamma \text{ fresh}) \\
\simeq_{\tau} & [\beta]\,\lambda x.\mu\gamma.[\alpha']\,(\mu\alpha.[\gamma]\,\mathsf{e}(\mathtt{LTC}'\langle[\alpha]\,t\rangle)) :: \mathsf{e}(s) & (\text{Lemma 6.2:3}) \\
= & [\beta]\,\lambda x.\mu\gamma.\mathsf{e}(([\gamma]\,\mathtt{LTC}'\langle[\alpha]\,t\rangle)[\![\alpha \setminus^{\alpha'} s]\!]) & \\
\simeq_{\tau} & [\beta]\,\lambda x.\mu\gamma.\mathsf{e}([\gamma]\,\mathtt{LTC}'\langle[\alpha']\,t :: s\rangle) & (\textit{i.h.}) \\
= & [\beta]\,\lambda x.\mu\gamma.[\gamma]\,\mathsf{e}(\mathtt{LTC}'\langle[\alpha']\,t :: s\rangle) & \\
\simeq_{\tau_7} & [\beta]\,\lambda x.\mathsf{e}(\mathtt{LTC}'\langle[\alpha']\,t :: s\rangle) & \\
= & \mathsf{e}([\beta]\,\lambda x.\mathtt{LTC}'\langle[\alpha']\,t :: s\rangle) & \\
= & \mathsf{e}(\mathtt{LCC}\langle[\alpha']\,t :: s\rangle) &
\end{array}
$$

Note that the *i.h.* applies since $\mathsf{sz}([\gamma]\,\mathtt{LTC}') < \mathsf{sz}(\mathtt{LCC})$.

- $\mathtt{LTC} = \mathtt{LTC}'[x \setminus u]$.

$$
\begin{array}{lll}
 & \mathsf{e}(([\beta]\,\mathtt{LTC}'\langle[\alpha]\,t\rangle[x \setminus t])[\![\alpha \setminus^{\alpha'} s]\!]) & \\
= & [\alpha']\,(\mu\alpha.\mathsf{e}([\beta]\,\mathtt{LTC}'\langle[\alpha]\,t\rangle[x \setminus t])) :: \mathsf{e}(s) & \\
= & [\alpha']\,(\mu\alpha.[\beta]\,(\lambda x.\mathsf{e}(\mathtt{LTC}'\langle[\alpha]\,t\rangle))\,\mathsf{e}(u)) :: \mathsf{e}(s) & \\
\simeq_{\tau_7} & [\alpha']\,(\mu\alpha.[\beta]\,(\mu\gamma.[\gamma]\,\lambda x.\mathsf{e}(\mathtt{LTC}'\langle[\alpha]\,t\rangle))\,\mathsf{e}(u)) :: \mathsf{e}(s) & (\gamma \text{ fresh}) \\
\simeq_{\tau} & [\beta]\,(\mu\gamma.[\alpha']\,(\mu\alpha.[\gamma]\,\lambda x.\mathsf{e}(\mathtt{LTC}'\langle[\alpha]\,t\rangle)) :: \mathsf{e}(s))\,\mathsf{e}(u) & (\text{Lemma 6.2:2}) \\
= & [\beta]\,(\mu\gamma.\mathsf{e}(([\gamma]\,\lambda x.\mathtt{LTC}'\langle[\alpha]\,t\rangle)[\![\alpha \setminus^{\alpha'} s]\!]))\,\mathsf{e}(u) & \\
\simeq_{\tau} & [\beta]\,(\mu\gamma.\mathsf{e}([\gamma]\,\lambda x.\mathtt{LTC}'\langle[\alpha']\,t :: s\rangle))\,\mathsf{e}(u) & (\textit{i.h.}) \\
= & [\beta]\,(\mu\gamma.[\gamma]\,\lambda x.\mathsf{e}(\mathtt{LTC}'\langle[\alpha']\,t :: s\rangle))\,\mathsf{e}(u) & \\
\simeq_{\tau_7} & [\beta]\,(\lambda x.\mathsf{e}(\mathtt{LTC}'\langle[\alpha']\,t :: s\rangle))\,\mathsf{e}(u) & \\
= & \mathsf{e}([\beta]\,\mathtt{LTC}'\langle[\alpha']\,t :: s\rangle[x \setminus u]) & \\
= & \mathsf{e}(\mathtt{LCC}\langle[\alpha']\,t :: s\rangle) &
\end{array}
$$

Note that the *i.h.* applies since $\mathsf{sz}([\gamma]\,\lambda x.\mathtt{LTC}') < \mathsf{sz}(\mathtt{LCC})$.

- $\mathtt{LCC} = \mathtt{LCC}'[\![\gamma \backslash^{\gamma'} s']\!]$.

$$\begin{array}{cll}
& \mathsf{e}((\mathtt{LCC}'\langle[\alpha]\,t\rangle[\![\gamma \backslash^{\gamma'} s']\!])[\![\alpha \backslash^{\alpha'} s]\!]) & \\
= & [\alpha']\,(\mu\alpha.\mathsf{e}(\mathtt{LCC}'\langle[\alpha]\,t\rangle[\![\gamma \backslash^{\gamma'} s']\!])) :: \mathsf{e}(s) & \\
= & [\alpha']\,(\mu\alpha.[\gamma']\,(\mu\gamma.\mathsf{e}(\mathtt{LCC}'\langle[\alpha]\,t\rangle)) :: \mathsf{e}(s')) :: \mathsf{e}(s) & \\
\simeq_\tau & [\gamma']\,(\mu\gamma.[\alpha']\,(\mu\alpha.\mathsf{e}(\mathtt{LCC}'\langle[\alpha]\,t\rangle)) :: \mathsf{e}(s)) :: \mathsf{e}(s') & \text{(Lemma 6.2:2)} \\
= & [\gamma']\,(\mu\gamma.\mathsf{e}(\mathtt{LCC}'\langle[\alpha]\,t\rangle[\![\alpha \backslash^{\alpha'} s]\!])) :: \mathsf{e}(s') & \\
\simeq_\tau & [\gamma']\,(\mu\gamma.\mathsf{e}(\mathtt{LCC}'\langle[\alpha']\,t :: s\rangle)) :: \mathsf{e}(s') & (i.h.) \\
= & \mathsf{e}(\mathtt{LCC}'\langle[\alpha']\,t :: s\rangle[\![\gamma \backslash^{\gamma'} s']\!]) & \\
= & \mathsf{e}(\mathtt{LCC}\langle[\alpha']\,t :: s\rangle) &
\end{array}$$

(2) We now address item (2), also by induction on the size of $\mathtt{LCC}$.

- $\mathtt{LCC} = \boxdot$.

$$\begin{array}{rcl}
\mathsf{e}(c[\![\beta \backslash^{\alpha} s']\!][\![\alpha \backslash^{\alpha'} s]\!]) & = & [\alpha']\,(\mu\alpha.[\alpha]\,(\mu\beta.\mathsf{e}(c)) :: \mathsf{e}(s')) :: \mathsf{e}(s) \\
& \simeq_{\tau_7} & [\alpha']\,(\mu\beta.\mathsf{e}(c)) :: \mathsf{e}(s') :: \mathsf{e}(s) \\
& = & \mathsf{e}(c[\![\beta \backslash^{\alpha'} s' \cdot s]\!])
\end{array}$$

- $\mathtt{LCC} = [\delta]\,\mathtt{LTC}$. We proceed by analyzing the shape of $\mathtt{LTC}$.
  - $\mathtt{LTC} = \mu\gamma.\mathtt{LCC}'$.

$$\begin{array}{cll}
& \mathsf{e}(([\delta]\,\mu\gamma.\mathtt{LCC}'\langle c[\![\beta \backslash^{\alpha} s']\!]\rangle)[\![\alpha \backslash^{\alpha'} s]\!]) & \\
= & [\alpha']\,(\mu\alpha.\mathsf{e}([\delta]\,\mu\gamma.\mathtt{LCC}'\langle c[\![\beta \backslash^{\alpha} s']\!]\rangle)) :: \mathsf{e}(s) & \\
= & [\alpha']\,(\mu\alpha.[\delta]\,\mu\gamma.\mathsf{e}(\mathtt{LCC}'\langle c[\![\beta \backslash^{\alpha} s']\!]\rangle)) :: \mathsf{e}(s) & \\
\simeq_\tau & [\delta]\,\mu\gamma.[\alpha']\,(\mu\alpha.\mathsf{e}(\mathtt{LCC}'\langle c[\![\beta \backslash^{\alpha} s']\!]\rangle)) :: \mathsf{e}(s) & \text{(Lemma 6.2:4)} \\
= & [\delta]\,\mu\gamma.\mathsf{e}(\mathtt{LCC}'\langle c[\![\beta \backslash^{\alpha} s']\!]\rangle[\![\alpha \backslash^{\alpha'} s]\!]) & \\
\simeq_\tau & [\delta]\,\mu\gamma.\mathsf{e}(\mathtt{LCC}'\langle c[\![\beta \backslash^{\alpha'} s' \cdot s]\!]\rangle) & (i.h.) \\
= & \mathsf{e}([\delta]\,\mu\gamma.\mathtt{LCC}'\langle c[\![\beta \backslash^{\alpha'} s' \cdot s]\!]\rangle) & \\
= & \mathsf{e}(\mathtt{LCC}\langle c[\![\beta \backslash^{\alpha'} s' \cdot s]\!]\rangle) &
\end{array}$$

  - $\mathtt{LTC} = \mathtt{LTC}'\,w$.

$$\begin{array}{cll}
& \mathsf{e}(([\delta]\,\mathtt{LTC}'\langle c[\![\beta \backslash^{\alpha} s']\!]\rangle\,w)[\![\alpha \backslash^{\alpha'} s]\!]) & \\
= & [\alpha']\,(\mu\alpha.[\delta]\,\mathsf{e}(\mathtt{LTC}'\langle c[\![\beta \backslash^{\alpha} s']\!]\rangle\,w)) :: \mathsf{e}(s) & \\
\simeq_{\tau_7} & [\alpha']\,(\mu\alpha.[\delta]\,(\mu\gamma.[\gamma]\,\mathsf{e}(\mathtt{LTC}'\langle c[\![\beta \backslash^{\alpha} s']\!]\rangle))\,\mathsf{e}(w)) :: \mathsf{e}(s) & (\gamma \text{ fresh}) \\
\simeq_\tau & [\delta]\,(\mu\gamma.[\alpha']\,(\mu\alpha.[\gamma]\,\mathsf{e}(\mathtt{LTC}'\langle c[\![\beta \backslash^{\alpha} s']\!]\rangle)) :: \mathsf{e}(s))\,\mathsf{e}(w) & \text{(Lemma 6.2:2)} \\
= & [\delta]\,(\mu\gamma.\mathsf{e}(([\gamma]\,\mathtt{LTC}'\langle c[\![\beta \backslash^{\alpha} s']\!]\rangle)[\![\alpha \backslash^{\alpha'} s]\!]))\,\mathsf{e}(w) & \\
\simeq_\tau & [\delta]\,(\mu\gamma.\mathsf{e}([\gamma]\,\mathtt{LTC}'\langle c[\![\beta \backslash^{\alpha'} s' \cdot s]\!]\rangle))\,\mathsf{e}(w) & (i.h.) \\
= & [\delta]\,(\mu\gamma.[\gamma]\,\mathsf{e}(\mathtt{LTC}'\langle c[\![\beta \backslash^{\alpha'} s' \cdot s]\!]\rangle))\,\mathsf{e}(w) & \\
\simeq_{\tau_7} & [\delta]\,(\mathsf{e}(\mathtt{LTC}'\langle c[\![\beta \backslash^{\alpha'} s' \cdot s]\!]\rangle))\,\mathsf{e}(w) & \\
= & \mathsf{e}([\delta]\,\mathtt{LTC}'\langle c[\![\beta \backslash^{\alpha'} s' \cdot s]\!]\rangle\,w) & \\
= & \mathsf{e}(\mathtt{LCC}\langle c[\![\beta \backslash^{\alpha'} s' \cdot s]\!]\rangle) &
\end{array}$$

Note that the *i.h.* applies since $\mathsf{sz}([\gamma]\,\mathtt{LTC}') < \mathsf{sz}(\mathtt{LCC})$.

– $\mathtt{LTC} = \lambda x.\mathtt{LTC}'$.

$$\begin{array}{lll} & \mathtt{e}(([\delta]\,\lambda x.\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle)[\![\alpha \setminus^{\alpha'} s]\!]) & \\ = & [\alpha']\,(\mu\alpha.[\delta]\,\mathtt{e}(\lambda x.\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle)) :: \mathtt{e}(s) & \\ \simeq_{\tau_7} & [\alpha']\,(\mu\alpha.[\delta]\,\lambda x.\mu\gamma.[\gamma]\,\mathtt{e}(\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle)) :: \mathtt{e}(s) & (\gamma \text{ fresh}) \\ \simeq_{\tau} & [\delta]\,\lambda x.\mu\gamma.[\alpha']\,(\mu\alpha.[\gamma]\,\mathtt{e}(\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle)) :: \mathtt{e}(s) & (\text{Lemma 6.2:3}) \\ = & [\delta]\,\lambda x.\mu\gamma.\mathtt{e}(([\gamma]\,\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle)[\![\alpha \setminus^{\alpha'} s]\!]) & \\ \simeq_{\tau} & [\delta]\,\lambda x.\mu\gamma.\mathtt{e}([\gamma]\,\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle) & (i.h.) \\ = & [\delta]\,\lambda x.\mu\gamma.[\gamma]\,\mathtt{e}(\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle) & \\ \simeq_{\tau_7} & [\delta]\,\lambda x.\mathtt{e}(\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle) & \\ = & \mathtt{e}([\delta]\,\lambda x.\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle) & \\ = & \mathtt{e}(\mathtt{LCC}\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle) & \end{array}$$

Note that the *i.h.* applies since $\mathtt{sz}([\gamma]\,\mathtt{LTC}') < \mathtt{sz}(\mathtt{LCC})$.

– $\mathtt{LTC} = \mathtt{LTC}'[x \setminus u]$.

$$\begin{array}{lll} & \mathtt{e}(([\delta]\,\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle[x \setminus u])[\![\alpha \setminus^{\alpha'} s]\!]) & \\ = & [\alpha']\,(\mu\alpha.[\delta]\,\mathtt{e}(\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle[x \setminus u])) :: \mathtt{e}(s) & \\ = & [\alpha']\,(\mu\alpha.[\delta]\,(\lambda x.\mathtt{e}(\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle))\,\mathtt{e}(u)) :: \mathtt{e}(s) & \\ \simeq_{\tau_7} & [\alpha']\,(\mu\alpha.[\delta]\,(\mu\gamma.[\gamma]\,\lambda x.\mathtt{e}(\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle))\,\mathtt{e}(u)) :: \mathtt{e}(s) & (\gamma \text{ fresh}) \\ \simeq_{\tau} & [\delta]\,(\mu\gamma.[\alpha']\,(\mu\alpha.[\gamma]\,\lambda x.\mathtt{e}(\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle)) :: \mathtt{e}(s))\,\mathtt{e}(u) & (\text{Lemma 6.2:2}) \\ = & [\delta]\,(\mu\gamma.\mathtt{e}(([\gamma]\,\lambda x.\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle)[\![\alpha \setminus^{\alpha'} s]\!]))\,\mathtt{e}(u) & \\ \simeq_{\tau} & [\delta]\,(\mu\gamma.\mathtt{e}([\gamma]\,\lambda x.\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle))\,\mathtt{e}(u) & (i.h.) \\ = & [\delta]\,(\mu\gamma.[\gamma]\,\lambda x.\mathtt{e}(\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle))\,\mathtt{e}(u) & \\ \simeq_{\tau_7} & [\delta]\,(\lambda x.\mathtt{e}(\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle))\,\mathtt{e}(u) & \\ = & \mathtt{e}([\delta]\,\mathtt{LTC}'\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle[x \setminus u]) & \\ = & \mathtt{e}(\mathtt{LCC}\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle) & \end{array}$$

Note that the *i.h.* applies since $\mathtt{sz}([\gamma]\,\lambda x.\mathtt{LTC}') < \mathtt{sz}(\mathtt{LCC})$.

- $\mathtt{LCC} = \mathtt{LCC}'[\![\gamma \setminus^{\gamma'} s'']\!]$.

$$\begin{array}{lll} & \mathtt{e}(\mathtt{LCC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\gamma \setminus^{\gamma'} s'']\!][\![\alpha \setminus^{\alpha'} s]\!]) & \\ = & [\alpha']\,(\mu\alpha.\mathtt{e}(\mathtt{LCC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\gamma \setminus^{\gamma'} s'']\!])) :: \mathtt{e}(s) & \\ = & [\alpha']\,(\mu\alpha.[\gamma']\,(\mu\gamma.\mathtt{e}(\mathtt{LCC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle)) :: \mathtt{e}(s'')) :: \mathtt{e}(s) & \\ \simeq_{\tau} & [\gamma']\,(\mu\gamma.[\alpha']\,(\mu\alpha.\mathtt{e}(\mathtt{LCC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle)) :: \mathtt{e}(s)) :: \mathtt{e}(s'') & (\text{Lemma 6.2:2}) \\ = & [\gamma']\,(\mu\gamma.\mathtt{e}(\mathtt{LCC}'\langle c[\![\beta \setminus^{\alpha} s']\!]\rangle[\![\alpha \setminus^{\alpha'} s]\!])) :: \mathtt{e}(s'') & \\ \simeq_{\tau} & [\gamma']\,(\mu\gamma.\mathtt{e}(\mathtt{LCC}'\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle)) :: \mathtt{e}(s'') & (i.h.) \\ = & \mathtt{e}(\mathtt{LCC}'\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle[\![\gamma \setminus^{\gamma'} s'']\!]) & \\ = & \mathtt{e}(\mathtt{LCC}\langle c[\![\beta \setminus^{\alpha'} s' \cdot s]\!]\rangle) & \end{array}$$

□

Lemma 6.7. *Let $o, p \in \mathbb{O}_{\lambda\mu}$. Then, $o \simeq_\tau p$ implies $\mathcal{P}(o) \simeq \mathcal{P}(p)$.*

Proof. By induction on $\simeq_\tau$. We first analyse the base cases.

- $o = (\lambda y.\lambda x.t)\,v \simeq_\tau \lambda x.(\lambda y.t)\,v = p$, where $x \notin v$. Then $\mathcal{P}(o) = \mathcal{P}((\lambda x.t)[y \setminus v]) \simeq \mathcal{P}(\lambda x.t[y \setminus v]) = \mathcal{P}(p)$ holds by Lemma 5.6:1.
- $o = (\lambda x.t\,v)\,u \simeq_\tau (\lambda x.t)\,u\,v = p$, where $x \notin v$. Then $\mathcal{P}(o) = \mathcal{P}((t\,u)[y \setminus v]) \simeq \mathcal{P}(t[y \setminus v]\,u) = \mathcal{P}(p)$ holds by Lemma 5.6:1.
- $o = (\lambda x.\mu\alpha.[\beta]\,u)\,v \simeq_\tau \mu\alpha.[\beta]\,(\lambda x.u)\,v = p$, where $\alpha \notin v$. Then $\mathcal{P}(o) = \mathcal{P}((\mu\alpha.[\beta]\,u)[x \setminus v]) \simeq \mathcal{P}(\mu\alpha.[\beta]\,u[x \setminus v]) = \mathcal{P}(p)$ holds by Lemma 5.6:1.

- $o = [\alpha'] \, (\mu\alpha.[\beta'] \, (\mu\beta.c) \, w) \, v \simeq_\tau [\beta'] \, (\mu\beta.[\alpha'] \, (\mu\alpha.c) \, v) \, w = p$, where $\alpha \notin w, \beta \notin v, \beta \neq \alpha', \alpha \neq \beta'$. Then $\mathcal{P}(o) = \mathcal{P}([\alpha'] \, \mu\alpha''.([\beta'] \, \mu\beta''.c[\![\beta \setminus^{\beta''} w]\!])[\![\alpha \setminus^{\alpha''} v]\!]) \simeq$ $\mathcal{P}([\beta'] \, \mu\beta''.([\alpha'] \, \mu\alpha''.c[\![\alpha \setminus^{\alpha''} v]\!])[\![\beta \setminus^{\beta''} w]\!]) = \mathcal{P}(p)$ holds by Lemma 5.6:2.
- $o = [\alpha'] \, (\mu\alpha.[\beta'] \, \lambda x.\mu\beta.c) \, v \simeq_\tau [\beta'] \, \lambda x.\mu\beta.[\alpha'] \, (\mu\alpha.c) \, v = p$, where $x \notin v, \beta \notin v, \beta \neq \alpha', \alpha \neq \beta'$. Then $\mathcal{P}(o) = \mathcal{P}([\alpha'] \, \mu\alpha''.([\beta'] \, \lambda x.\mu\beta.c)[\![\alpha \setminus^{\alpha''} v]\!]) \simeq \mathcal{P}([\beta'] \, \lambda x.\mu\beta.[\alpha'] \, \mu\alpha''.c[\![\alpha \setminus^{\alpha''} v]\!]) = \mathcal{P}(p)$ holds by Lemma 5.6:2.
- $o = [\alpha'] \, \lambda x.\mu\alpha.[\beta'] \, \lambda y.\mu\beta.c \simeq_\tau [\beta'] \, \lambda y.\mu\beta.[\alpha'] \, \lambda x.\mu\alpha.c = p$, where $\beta \neq \alpha', \alpha \neq \beta'$. Then $\mathcal{P}(o) = [\alpha'] \, \lambda x.\mu\alpha.[\beta'] \, \lambda y.\mu\beta.\mathcal{P}(c) \simeq_{\mathtt{ppop}} [\beta'] \, \lambda y.\mu\beta.[\alpha'] \, \lambda x.\mu\alpha.\mathcal{P}(c) = \mathcal{P}(p)$.
- $o = \mu\alpha.[\alpha] \, v \simeq_\tau v$, where $\alpha \notin v$. Then $\mathcal{P}(o) = \mu\alpha.[\alpha] \, \mathcal{P}(v) \simeq_\theta \mathcal{P}(v) = \mathcal{P}(p)$.
- $o = [\alpha'] \, \mu\alpha.[\beta'] \, \mu\beta.c \simeq_\tau [\beta'] \, \mu\beta.[\alpha'] \, \mu\alpha.c = p$, where $\beta' \neq \alpha$ and $\alpha' \neq \beta$. Then $\mathcal{P}(o) = [\alpha'] \, \mu\alpha.[\beta'] \, \mu\beta.\mathcal{P}(c) \simeq_{\mathtt{exren}} [\beta'] \, \mu\beta.[\alpha'] \, \mu\alpha.\mathcal{P}(c) = \mathcal{P}(p)$.
- $o = [\alpha'] \, \mu\alpha.[\beta'] \, \lambda x.\mu\beta.c \simeq_\tau [\beta'] \, \lambda x.\mu\beta.[\alpha'] \, \mu\alpha.c = p$, where $\beta' \neq \alpha$ and $\alpha' \neq \beta$. Then $\mathcal{P}(o) = [\alpha'] \, \mu\alpha.[\beta'] \, \lambda x.\mu\beta.\mathcal{P}(c) \simeq_{\mathtt{exren}} [\beta'] \, \lambda x.\mu\beta.[\alpha'] \, \mu\alpha.\mathcal{P}(c) = \mathcal{P}(p)$.
- $o = [\alpha'] \, \mu\alpha.[\beta'] \, (\mu\beta.c)w \simeq_\tau [\beta'] \, (\mu\beta.[\alpha'] \, \mu\alpha.c)w = p$, where $\beta' \neq \alpha$ and $\alpha' \neq \beta$. Then $\mathcal{P}(o) = \mathcal{P}([\alpha'] \, \mu\alpha.[\beta'] \, \mu\beta''.c[\![\beta \setminus^{\beta''} w]\!]) \simeq \mathcal{P}([\beta'] \, \mu\beta''.([\alpha'] \, \mu\alpha.c)[\![\beta \setminus^{\beta''} w]\!]) = \mathcal{P}(p)$ holds by Lemma 5.6:2.

For the inductive cases, let consider $o = \mathtt{O}\langle o' \rangle \simeq_\tau \mathtt{O}\langle p' \rangle = p$, where $o' \simeq_\tau p'$. The *i.h.* gives $\mathcal{P}(o') \simeq \mathcal{P}(p')$, and Lemma 5.7 gives $\mathcal{P}(\mathtt{O}\langle \mathcal{P}(o') \rangle) \simeq \mathcal{P}(\mathtt{O}\langle \mathcal{P}(p') \rangle)$, so that we conclude by the fact that $\mathcal{P}(\mathtt{O}\langle \mathcal{P}(o') \rangle) = \mathcal{P}(\mathtt{O}\langle o' \rangle)$ and $\mathcal{P}(\mathtt{O}\langle \mathcal{P}(p') \rangle) = \mathcal{P}(\mathtt{O}\langle p' \rangle)$. □

Lemma 6.8. *Let $o, p \in \mathbb{O}_{\Lambda M}$. Then, $o \simeq p$ implies $\mathsf{e}(o) \simeq_\tau \mathsf{e}(p)$.*

Proof. The cases where $o \simeq p$ holds by reflexivity, transitivity or symmetry are straightforward. For congruence, we reason by induction on the context $\mathtt{O}$ such that $o = \mathtt{O}\langle l \rangle$ and $p = \mathtt{O}\langle r \rangle$ with $l \simeq_* r$, where $\simeq_*$ is an axiom in Figure 5.

- $\mathtt{O} = \Box$. There are two possible rules:
  1. $\simeq_{\mathtt{exsubs}}$. Then, $o = \mathtt{LTT}\langle t \rangle[x \setminus u]$ and $p = \mathtt{LTT}\langle t[x \setminus u] \rangle$ with $x \notin \mathtt{LTT}$ and $\mathsf{fc}(u, \mathtt{LTT})$. We proceed by induction on the size of $\mathtt{LTT}$:
     - $\mathtt{LTT} = \Box$. This case is immediate since $o = p$.
     - $\mathtt{LTT} = \mathtt{LTT}' \, v$. Then,

$$\begin{array}{rcll} \mathsf{e}((\mathtt{LTT}'\langle t \rangle \, v)[x \setminus u]) & = & (\lambda x.\mathsf{e}(\mathtt{LTT}'\langle t \rangle) \, \mathsf{e}(v)) \, \mathsf{e}(u) & \\ & \simeq_{\tau_2} & (\lambda x.\mathsf{e}(\mathtt{LTT}'\langle t \rangle)) \, \mathsf{e}(u) \, \mathsf{e}(v) & \\ & = & \mathsf{e}(\mathtt{LTT}'\langle t \rangle[x \setminus u]) \, \mathsf{e}(v) & \\ & \simeq_\tau & \mathsf{e}(\mathtt{LTT}'\langle t[x \setminus u] \rangle) \, \mathsf{e}(v) & (i.h.) \\ & = & \mathsf{e}(\mathtt{LTT}'\langle t[x \setminus u] \rangle \, v) & \end{array}$$

     - $\mathtt{LTT} = \lambda y.\mathtt{LTT}'$. Then,

$$\begin{array}{rcll} \mathsf{e}((\lambda y.\mathtt{LTT}'\langle t \rangle)[x \setminus u]) & = & (\lambda x.\lambda y.\mathsf{e}(\mathtt{LTT}'\langle t \rangle)) \, \mathsf{e}(u) & \\ & \simeq_{\tau_1} & \lambda y.(\lambda x.\mathsf{e}(\mathtt{LTT}'\langle t \rangle)) \, \mathsf{e}(u) & \\ & = & \lambda y.\mathsf{e}(\mathtt{LTT}'\langle t \rangle[x \setminus u]) & \\ & \simeq_\tau & \lambda y.\mathsf{e}(\mathtt{LTT}'\langle t[x \setminus u] \rangle) & (i.h.) \\ & = & \mathsf{e}(\lambda y.\mathtt{LTT}'\langle t[x \setminus u] \rangle) & \end{array}$$

     - $\mathtt{LTT} = \mu\alpha.\mathtt{LCT}$. We proceed by analyzing the shape of $\mathtt{LCT}$:

∗ $\texttt{LCT} = [\delta]\,\texttt{LTT}'$. Then,

$$\begin{array}{rcll} \mathsf{e}((\mu\alpha.[\delta]\,\texttt{LTT}'\langle t\rangle)[x\setminus u]) & = & (\lambda x.\mu\alpha.[\delta]\,\mathsf{e}(\texttt{LTT}'\langle t\rangle))\,\mathsf{e}(u) & \\ & \simeq_{\tau_3} & \mu\alpha.[\delta]\,(\lambda x.\mathsf{e}(\texttt{LTT}'\langle t\rangle))\,\mathsf{e}(u) & \\ & = & \mu\alpha.[\delta]\,\mathsf{e}(\texttt{LTT}'\langle t\rangle[x\setminus u]) & \\ & \simeq_{\tau} & \mu\alpha.[\delta]\,\mathsf{e}(\texttt{LTT}'\langle t[x\setminus u]\rangle) & (\textit{i.h.}) \\ & = & \mathsf{e}(\mu\alpha.[\delta]\,\texttt{LTT}'\langle t[x\setminus u]\rangle) & \end{array}$$

∗ $\texttt{LCT} = \texttt{LCT}'[\![\gamma \setminus^{\delta} s]\!]$. Then,

$$\begin{array}{rcll} \mathsf{e}((\mu\alpha.\texttt{LCT}'\langle t\rangle[\![\gamma\setminus^{\delta} s]\!])[x\setminus u]) & = & (\lambda x.\mu\alpha.[\delta]\,(\mu\gamma.\mathsf{e}(\texttt{LCT}'\langle t\rangle)) :: \mathsf{e}(s))\,\mathsf{e}(u) & \\ & \simeq_{\tau_3} & \mu\alpha.[\delta]\,(\lambda x.(\mu\gamma.\mathsf{e}(\texttt{LCT}'\langle t\rangle)) :: \mathsf{e}(s))\,\mathsf{e}(u) & \\ & \simeq_{\tau} & \mu\alpha.[\delta]\,((\lambda x.\mu\gamma.\mathsf{e}(\texttt{LCT}'\langle t\rangle))\,\mathsf{e}(u)) :: \mathsf{e}(s) & \text{(L. 6.2:1)} \\ & = & \mu\alpha.[\delta]\,\mathsf{e}((\mu\gamma.\texttt{LCT}'\langle t\rangle)[x\setminus u]) :: \mathsf{e}(s) & \\ & \simeq_{\tau} & \mu\alpha.[\delta]\,\mathsf{e}(\mu\gamma.\texttt{LCT}'\langle t[x\setminus u]\rangle) :: \mathsf{e}(s) & (\textit{i.h.}) \\ & = & \mu\alpha.[\delta]\,(\mu\gamma.\mathsf{e}(\texttt{LCT}'\langle t[x\setminus u]\rangle)) :: \mathsf{e}(s) & \\ & = & \mu\alpha.\mathsf{e}(\texttt{LCT}'\langle t[x\setminus u]\rangle[\![\gamma\setminus^{\delta} s]\!]) & \\ & = & \mathsf{e}(\mu\alpha.\texttt{LCT}'\langle t[x\setminus u]\rangle[\![\gamma\setminus^{\delta} s]\!]) & \end{array}$$

– $\texttt{LTT} = \texttt{LTT}'[y\setminus v]$. Then,

$$\begin{array}{rcll} \mathsf{e}(\texttt{LTT}'\langle t\rangle[y\setminus v][x\setminus u]) & = & (\lambda x.(\lambda y.\mathsf{e}(\texttt{LTT}'\langle t\rangle))\,\mathsf{e}(v))\,\mathsf{e}(u) & \\ & \simeq_{\tau_2} & (\lambda x.\lambda y.\mathsf{e}(\texttt{LTT}'\langle t\rangle))\,\mathsf{e}(u)\,\mathsf{e}(v) & \\ & \simeq_{\tau_1} & (\lambda y.(\lambda x.\mathsf{e}(\texttt{LTT}'\langle t\rangle))\,\mathsf{e}(u))\,\mathsf{e}(v) & \\ & = & (\lambda y.\mathsf{e}(\texttt{LTT}'\langle t\rangle[x\setminus u]))\,\mathsf{e}(v) & \\ & \simeq_{\tau} & (\lambda y.\mathsf{e}(\texttt{LTT}'\langle t[x\setminus u]\rangle))\,\mathsf{e}(v) & (\textit{i.h.}) \\ & = & \mathsf{e}(\texttt{LTT}'\langle t[x\setminus u]\rangle[y\setminus v]) & \end{array}$$

(2) $\simeq_{\theta}$. Then, $o = \mu\alpha.[\alpha]\,t$ and $p = t$ with $\alpha \notin t$. Moreover, $\mathsf{e}(\mu\alpha.[\alpha]\,t) = \mu\alpha.[\alpha]\,\mathsf{e}(t)$ by definition. We conclude by $\simeq_{\tau_7}$.

- $\texttt{O} = \boxdot$. There are three possible rules:
  (1) $\simeq_{\texttt{exrepl}}$. Then, $o = \texttt{LCC}\langle c\rangle[\![\alpha\setminus^{\beta} s]\!]$ and $p = \texttt{LCC}\langle c[\![\alpha\setminus^{\beta} s]\!]\rangle$ with $\alpha \notin \texttt{LCC}$, $\mathsf{fc}(\beta, \texttt{LCC})$ and $\mathsf{fc}(s, \texttt{LCC})$. We proceed by induction on the size of $\texttt{LCC}$:
  – $\texttt{LCC} = \boxdot$. This case is immediate since $o = p$.
  – $\texttt{LCC} = [\delta]\,\texttt{LTC}$. We proceed by analyzing the shape of $\texttt{LTC}$:
  ∗ $\texttt{LTC}'\,u$. Then,

$$\begin{array}{rlll} & \mathsf{e}(([\delta]\,\texttt{LTC}'\langle c\rangle\,u)[\![\alpha\setminus^{\beta} s]\!]) & \\ = & [\beta]\,(\mu\alpha.[\delta]\,\mathsf{e}(\texttt{LTC}'\langle c\rangle)\,\mathsf{e}(u)) :: \mathsf{e}(s) & \\ \simeq_{\tau_7} & [\beta]\,(\mu\alpha.[\delta]\,(\mu\delta'.[\delta']\,\mathsf{e}(\texttt{LTC}'\langle c\rangle))\,\mathsf{e}(u)) :: \mathsf{e}(s) & (\delta' \text{ fresh}) \\ \simeq_{\tau} & [\delta]\,(\mu\delta'.[\beta]\,(\mu\alpha.[\delta']\,\mathsf{e}(\texttt{LTC}'\langle c\rangle)) :: \mathsf{e}(s))\,\mathsf{e}(u) & \text{(L. 6.2:2)} \\ = & [\delta]\,(\mu\delta'.\mathsf{e}(([\delta']\,\texttt{LTC}'\langle c\rangle)[\![\alpha\setminus^{\beta} s]\!]))\,\mathsf{e}(u) & \\ \simeq_{\tau} & [\delta]\,(\mu\delta'.\mathsf{e}([\delta']\,\texttt{LTC}'\langle c[\![\alpha\setminus^{\beta} s]\!]\rangle))\,\mathsf{e}(u) & (\textit{i.h.}) \\ = & [\delta]\,(\mu\delta'.[\delta']\,\mathsf{e}(\texttt{LTC}'\langle c[\![\alpha\setminus^{\beta} s]\!]\rangle))\,\mathsf{e}(u) & \\ \simeq_{\tau_7} & [\delta]\,\mathsf{e}(\texttt{LTC}'\langle c[\![\alpha\setminus^{\beta} s]\!]\rangle)\,\mathsf{e}(u) & \\ = & \mathsf{e}([\delta]\,\texttt{LTC}'\langle c[\![\alpha\setminus^{\beta} s]\!]\rangle\,u) & \end{array}$$

$*$ $\lambda x.\mathtt{LTC}'$. Then,

$$\begin{array}{lll} & \mathsf{e}(([\delta]\,\lambda x.\mathtt{LTC}'\langle c\rangle)[\![\alpha \setminus^{\beta} s]\!]) & \\ = & [\beta]\,(\mu\alpha.[\delta]\,\lambda x.\mathsf{e}(\mathtt{LTC}'\langle c\rangle)) :: \mathsf{e}(s) & \\ \simeq_{\tau_7} & [\beta]\,(\mu\alpha.[\delta]\,\lambda x.\mu\delta'.[\delta']\,\mathsf{e}(\mathtt{LTC}'\langle c\rangle)) :: \mathsf{e}(s) & (\delta' \text{ fresh}) \\ \simeq_{\tau} & [\delta]\,\lambda x.\mu\delta'.[\beta]\,(\mu\alpha.[\delta']\,\mathsf{e}(\mathtt{LTC}'\langle c\rangle)) :: \mathsf{e}(s) & (\text{L. 6.2:3}) \\ = & [\delta]\,\lambda x.\mu\delta'.\mathsf{e}(([\delta']\,\mathtt{LTC}'\langle c\rangle)[\![\alpha \setminus^{\beta} s]\!]) & \\ \simeq_{\tau} & [\delta]\,\lambda x.\mu\delta'.\mathsf{e}([\delta']\,\mathtt{LTC}'\langle c[\![\alpha \setminus^{\beta} s]\!]\rangle) & (\textit{i.h.}) \\ = & [\delta]\,\lambda x.\mu\delta'.[\delta']\,\mathsf{e}(\mathtt{LTC}'\langle c[\![\alpha \setminus^{\beta} s]\!]\rangle) & \\ \simeq_{\tau_7} & [\delta]\,\lambda x.\mathsf{e}(\mathtt{LTC}'\langle c[\![\alpha \setminus^{\beta} s]\!]\rangle) & \\ = & \mathsf{e}([\delta]\,\lambda x.\mathtt{LTC}'\langle c[\![\alpha \setminus^{\beta} s]\!]\rangle) & \end{array}$$

$*$ $\mu\gamma.\mathtt{LCC}'$. Then,

$$\begin{array}{lll} & \mathsf{e}(([\delta]\,\mu\gamma.\mathtt{LCC}'\langle c\rangle)[\![\alpha \setminus^{\beta} s]\!]) & \\ = & [\beta]\,(\mu\alpha.[\delta]\,\mu\gamma.\mathsf{e}(\mathtt{LCC}'\langle c\rangle)) :: \mathsf{e}(s) & \\ \simeq_{\tau} & [\delta]\,\mu\gamma.[\beta]\,(\mu\alpha.\mathsf{e}(\mathtt{LCC}'\langle c\rangle)) :: \mathsf{e}(s) & (\text{L. 6.2:4}) \\ = & [\delta]\,\mu\gamma.\mathsf{e}((\mathtt{LCC}'\langle c\rangle)[\![\alpha \setminus^{\beta} s]\!]) & \\ \simeq_{\tau} & [\delta]\,\mu\gamma.\mathsf{e}(\mathtt{LCC}'\langle c[\![\alpha \setminus^{\beta} s]\!]\rangle) & (\textit{i.h.}) \\ = & \mathsf{e}([\delta]\,\mu\gamma.\mathtt{LCC}'\langle c[\![\alpha \setminus^{\beta} s]\!]\rangle) & \end{array}$$

$*$ $\mathtt{LTC}'[x \setminus u]$. Then,

$$\begin{array}{lll} & \mathsf{e}(([\delta]\,\mathtt{LTC}'\langle c\rangle[x \setminus u])[\![\alpha \setminus^{\beta} s]\!]) & \\ = & [\beta]\,(\mu\alpha.[\delta]\,(\lambda x.\mathsf{e}(\mathtt{LTC}'\langle c\rangle))\,\mathsf{e}(u)) :: \mathsf{e}(s) & \\ \simeq_{\tau_7} & [\beta]\,(\mu\alpha.[\delta]\,(\mu\delta'.[\delta']\,\lambda x.\mathsf{e}(\mathtt{LTC}'\langle c\rangle))\,\mathsf{e}(u)) :: \mathsf{e}(s) & (\delta' \text{ fresh}) \\ \simeq_{\tau} & [\delta]\,(\mu\delta'.[\beta]\,(\mu\alpha.[\delta']\,\lambda x.\mathsf{e}(\mathtt{LTC}'\langle c\rangle)) :: \mathsf{e}(s))\,\mathsf{e}(u) & (\text{L. 6.2:2}) \\ = & [\delta]\,(\mu\delta'.\mathsf{e}(([\delta']\,\lambda x.\mathtt{LTC}'\langle c\rangle)[\![\alpha \setminus^{\beta} s]\!]))\,\mathsf{e}(u) & \\ \simeq_{\tau} & [\delta]\,(\mu\delta'.\mathsf{e}([\delta']\,\lambda x.\mathtt{LTC}'\langle c[\![\alpha \setminus^{\beta} s]\!]\rangle))\,\mathsf{e}(u) & (\textit{i.h.}) \\ = & [\delta]\,(\mu\delta'.[\delta']\,\mathsf{e}(\lambda x.\mathtt{LTC}'\langle c[\![\alpha \setminus^{\beta} s]\!]\rangle))\,\mathsf{e}(u) & \\ \simeq_{\tau_7} & [\delta]\,\mathsf{e}(\lambda x.\mathtt{LTC}'\langle c[\![\alpha \setminus^{\beta} s]\!]\rangle)\,\mathsf{e}(u) & \\ = & \mathsf{e}([\delta]\,\mathtt{LTC}'\langle c[\![\alpha \setminus^{\beta} s]\!]\rangle[x \setminus u]) & \end{array}$$

– $\mathtt{LCC} = \mathtt{LCC}'[\![\gamma \setminus^{\delta} s']\!]$. Then,

$$\begin{array}{lll} & \mathsf{e}(\mathtt{LTC}'\langle c\rangle[\![\gamma \setminus^{\delta} s']\!][\![\alpha \setminus^{\beta} s]\!]) & \\ = & [\beta]\,(\mu\alpha.[\delta]\,(\mu\gamma.\mathsf{e}(\mathtt{LTC}'\langle c\rangle)) :: \mathsf{e}(s')) :: \mathsf{e}(s) & \\ \simeq_{\tau} & [\delta]\,(\mu\gamma.[\beta]\,(\mu\alpha.\mathsf{e}(\mathtt{LTC}'\langle c\rangle)) :: \mathsf{e}(s)) :: \mathsf{e}(s') & (\text{L. 6.2:2}) \\ = & [\delta]\,(\mu\gamma.\mathsf{e}(\mathtt{LTC}'\langle c\rangle[\![\alpha \setminus^{\beta} s]\!])) :: \mathsf{e}(s') & \\ \simeq_{\tau} & [\delta]\,(\mu\gamma.\mathsf{e}(\mathtt{LTC}'\langle c[\![\alpha \setminus^{\beta} s]\!]\rangle)) :: \mathsf{e}(s') & (\textit{i.h.}) \\ = & \mathsf{e}(\mathtt{LTC}'\langle c\rangle[\![\alpha \setminus^{\beta} s]\!][\![\gamma \setminus^{\delta} s']\!]) & \end{array}$$

(2) $\simeq_{\mathtt{exren}}$. Then, $o = [\beta]\,\mu\alpha.\mathtt{LCC}\langle c\rangle$ and $p = \mathtt{LCC}\langle[\beta]\,\mu\alpha.c\rangle$ with $\alpha \notin \mathtt{LCC}$ and $\mathtt{fc}(\beta, \mathtt{LCC})$. We proceed by induction on the size of $\mathtt{LCC}$:
– $\mathtt{LCC} = \boxdot$. This case is immediate since $o = p$.
– $\mathtt{LCC} = [\delta]\,\mathtt{LTC}$. We proceed by analyzing the shape of $\mathtt{LTC}$:

* $\mathtt{LTC}'\,u$. Then,

$$\begin{array}{ll}
 & \mathsf{e}([\beta]\,\mu\alpha.[\delta]\,\mathtt{LTC}'\langle c\rangle\,u) \\
= & [\beta]\,\mu\alpha.[\delta]\,\mathsf{e}(\mathtt{LTC}'\langle c\rangle)\,\mathsf{e}(u) \\
\simeq_{\tau_7} & [\beta]\,\mu\alpha.[\delta]\,(\mu\delta'.[\delta']\,\mathsf{e}(\mathtt{LTC}'\langle c\rangle))\,\mathsf{e}(u) \quad (\delta' \text{ fresh}) \\
\simeq_{\tau_{10}} & [\delta]\,(\mu\delta'.[\beta]\,\mu\alpha.[\delta']\,\mathsf{e}(\mathtt{LTC}'\langle c\rangle))\,\mathsf{e}(u) \\
= & [\delta]\,(\mu\delta'.\mathsf{e}([\beta]\,\mu\alpha.[\delta']\,\mathtt{LTC}'\langle c\rangle))\,\mathsf{e}(u) \\
\simeq_{\tau} & [\delta]\,(\mu\delta'.\mathsf{e}([\delta']\,\mathtt{LTC}'\langle[\beta]\,\mu\alpha.c\rangle))\,\mathsf{e}(u) \quad (i.h.) \\
= & [\delta]\,(\mu\delta'.[\delta']\,\mathsf{e}(\mathtt{LTC}'\langle[\beta]\,\mu\alpha.c\rangle))\,\mathsf{e}(u) \\
\simeq_{\tau_7} & [\delta]\,\mathsf{e}(\mathtt{LTC}'\langle[\beta]\,\mu\alpha.c\rangle)\,\mathsf{e}(u) \\
= & \mathsf{e}([\delta]\,\mathtt{LTC}'\langle[\beta]\,\mu\alpha.c\rangle\,u)
\end{array}$$

* $\lambda x.\mathtt{LTC}'$. Then,

$$\begin{array}{ll}
 & \mathsf{e}([\beta]\,\mu\alpha.[\delta]\,\lambda x.\mathtt{LTC}'\langle c\rangle) \\
= & [\beta]\,\mu\alpha.[\delta]\,\lambda x.\mathsf{e}(\mathtt{LTC}'\langle c\rangle) \\
\simeq_{\tau_7} & [\beta]\,\mu\alpha.[\delta]\,\lambda x.\mu\delta'.[\delta']\,\mathsf{e}(\mathtt{LTC}'\langle c\rangle) \quad (\delta' \text{ fresh}) \\
\simeq_{\tau_9} & [\delta]\,\lambda x.\mu\delta'.[\beta]\,\mu\alpha.[\delta']\,\mathsf{e}(\mathtt{LTC}'\langle c\rangle) \\
= & [\delta]\,\lambda x.\mu\delta'.\mathsf{e}([\beta]\,\mu\alpha.[\delta']\,\mathtt{LTC}'\langle c\rangle) \\
\simeq_{\tau} & [\delta]\,\lambda x.\mu\delta'.\mathsf{e}([\delta']\,\mathtt{LTC}'\langle[\beta]\,\mu\alpha.c\rangle) \quad (i.h.) \\
= & [\delta]\,\lambda x.\mu\delta'.[\delta']\,\mathsf{e}(\mathtt{LTC}'\langle[\beta]\,\mu\alpha.c\rangle) \\
\simeq_{\tau_7} & [\delta]\,\lambda x.\mathsf{e}(\mathtt{LTC}'\langle[\beta]\,\mu\alpha.c\rangle) \\
= & \mathsf{e}([\delta]\,\lambda x.\mathtt{LTC}'\langle[\beta]\,\mu\alpha.c\rangle)
\end{array}$$

* $\mu\gamma.\mathtt{LCC}'$. Then,

$$\begin{array}{ll}
 & \mathsf{e}([\beta]\,\mu\alpha.[\delta]\,\mu\gamma.\mathtt{LCC}'\langle c\rangle) \\
= & [\beta]\,\mu\alpha.[\delta]\,\mu\gamma.\mathsf{e}(\mathtt{LCC}'\langle c\rangle) \\
\simeq_{\tau_8} & [\delta]\,\mu\gamma.[\beta]\,\mu\alpha.\mathsf{e}(\mathtt{LCC}'\langle c\rangle) \\
= & [\delta]\,\mu\gamma.\mathsf{e}([\beta]\,\mu\alpha.\mathtt{LCC}'\langle c\rangle) \\
\simeq_{\tau} & [\delta]\,\mu\gamma.\mathsf{e}(\mathtt{LCC}'\langle[\beta]\,\mu\alpha.c\rangle) \quad (i.h.) \\
= & [\delta]\,\mu\gamma.\mathsf{e}(\mathtt{LCC}'\langle[\beta]\,\mu\alpha.c\rangle) \\
= & \mathsf{e}([\delta]\,\mu\gamma.\mathtt{LCC}'\langle[\beta]\,\mu\alpha.c\rangle)
\end{array}$$

* $\mathtt{LTC}'[x \setminus u]$. Then,

$$\begin{array}{ll}
 & \mathsf{e}([\beta]\,\mu\alpha.[\delta]\,\mathtt{LTC}'\langle c\rangle[x \setminus u]) \\
= & [\beta]\,\mu\alpha.[\delta]\,(\lambda x.\mathsf{e}(\mathtt{LTC}'\langle c\rangle))\,\mathsf{e}(u) \\
\simeq_{\tau_7} & [\beta]\,\mu\alpha.[\delta]\,(\mu\delta'.[\delta']\,\lambda x.\mathsf{e}(\mathtt{LTC}'\langle c\rangle))\,\mathsf{e}(u) \quad (\delta' \text{ fresh}) \\
\simeq_{\tau_{10}} & [\delta]\,(\mu\delta'.[\beta]\,\mu\alpha.[\delta']\,\lambda x.\mathsf{e}(\mathtt{LTC}'\langle c\rangle))\,\mathsf{e}(u) \\
= & [\delta]\,(\mu\delta'.\mathsf{e}([\beta]\,\mu\alpha.[\delta']\,\lambda x.\mathtt{LTC}'\langle c\rangle))\,\mathsf{e}(u) \\
\simeq_{\tau} & [\delta]\,(\mu\delta'.\mathsf{e}([\delta']\,\lambda x.\mathtt{LTC}'\langle[\beta]\,\mu\alpha.c\rangle))\,\mathsf{e}(u) \quad (i.h.) \\
= & [\delta]\,(\mu\delta'.[\delta']\,\mathsf{e}(\lambda x.\mathtt{LTC}'\langle[\beta]\,\mu\alpha.c\rangle))\,\mathsf{e}(u) \\
\simeq_{\tau_7} & [\delta]\,\mathsf{e}(\lambda x.\mathtt{LTC}'\langle[\beta]\,\mu\alpha.c\rangle)\,\mathsf{e}(u) \\
= & \mathsf{e}([\delta]\,\mathtt{LTC}'\langle[\beta]\,\mu\alpha.c\rangle[x \setminus u])
\end{array}$$

– $\texttt{LCC} = \texttt{LCC}'[\![\gamma \setminus^{\delta} s']\!]$. Then,

$$\begin{array}{ll} & \mathsf{e}([\beta]\, \mu\alpha.\texttt{LCC}'\langle c\rangle[\![\gamma \setminus^{\delta} s']\!]) \\ = & [\beta]\, \mu\alpha.[\delta]\, (\mu\gamma.\mathsf{e}(\texttt{LCC}'\langle c\rangle)) :: \mathsf{e}(s') \\ \simeq_{\tau} & [\delta]\, (\mu\gamma.[\beta]\, \mu\alpha.\mathsf{e}(\texttt{LCC}'\langle c\rangle)) :: \mathsf{e}(s') \quad \text{(L. 6.2:4)} \\ = & [\delta]\, (\mu\gamma.\mathsf{e}([\beta]\, \mu\alpha.\texttt{LCC}'\langle c\rangle)) :: \mathsf{e}(s') \\ \simeq_{\tau} & [\delta]\, (\mu\gamma.\mathsf{e}(\texttt{LCC}'\langle [\beta]\, \mu\alpha.c\rangle)) :: \mathsf{e}(s') \quad (\textit{i.h.}) \\ = & \mathsf{e}(\texttt{LCC}'\langle [\beta]\, \mu\alpha.c\rangle[\![\gamma \setminus^{\delta} s']\!]) \end{array}$$

(3) $\simeq_{\mathtt{ppop}}$. Then, $o = [\alpha']\, \lambda x.\mu\alpha.[\beta']\, \lambda y.\mu\beta.c$ and $p = [\beta']\, \lambda y.\mu\beta.[\alpha']\, \lambda x.\mu\alpha.c$ with $\beta \neq \alpha'$ and $\alpha \neq \beta'$. Moreover, by definition we have $\mathsf{e}(o) = [\alpha']\, \lambda x.\mu\alpha.[\beta']\, \lambda y.\mu\beta.\mathsf{e}(c)$ and $\mathsf{e}(p) = [\beta']\, \lambda y.\mu\beta.[\alpha']\, \lambda x.\mu\alpha.\mathsf{e}(c)$. We conclude by $\simeq_{\tau_6}$.

- All the remaining cases are straightforward by using the *i.h.* □