A fully-abstract output-based semantics of $\lambda\mu$ in the π-calculus

Maria Grazia Vigliotti

Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2BZ, UK

Abstract

We study the $\lambda\mu$-calculus, extended with explicit substitution, and define a compositional output-based interpretation into a variant of the π-calculus with pairing. We show that this interpretation preserves single-step explicit head-reduction with respect to weak bisimilarity. We use this result to show operational soundness for head reduction, adequacy, and operational completeness. We also show that weak (*i.e.* lazy) reduction on closed terms is directly implemented through synchronisation.
We define four notions of weak equivalence for $\lambda\mu$ – one based on weak reduction '$\sim_{w\beta\mu}$', two modelling weak head-reduction and weak explicit head reduction, '$\sim_{w\mathrm{H}}$' and '$\sim_{w\mathrm{XH}}$' respectively (all considering terms without weak head-normal form equivalent as well), and one based on weak approximation '$\sim_{\mathcal{A}}$' – and show they all coincide. We will then show full abstraction results for our interpretation for the weak equivalences with respect to weak bisimilarity on processes using the approach of approximation.

Introduction

The relation between Church's pure λ-calculus [24, 15] and Milner's π-calculus [45] has been studied extensively in the past. Work started with Milner's encoding of the lazy λ-calculus[1] [43] and quickly led to more thorough investigations (see [44, 51, 21, 52, 12]; many more papers were written on the topic), also in the direction of object oriented calculi [35, 52]. The strength of the results that have been shown in those papers – like soundness, completeness, termination, and full abstraction – has encouraged researchers to investigate interpretations into the π-calculus of various calculi that have their foundation in classical logic, as done in, for example, [36, 10, 25, 17]. The particular calculus we focus on here is Parigot's $\lambda\mu$-calculus [48]; it is a natural extension of the λ-calculus by allowing names for terms as in $[\alpha]M$ and μ-abstractions over names through $\mu\alpha.M$, and contextual reduction rules that allow for the redirection of applicative contexts to named sub-terms.

Rather than looking at how to encode known control features into calculi like the λ-calculus, $\lambda\mu$, or $\Lambda\mu$[2] [32], as has been done in great detail by others, we focus on trying to understand what is exactly the notion of computation that is embedded in calculi like $\lambda\mu$; we approach that problem here by presenting a fully abstract interpretation for that calculus into the π-calculus. Perhaps the most unexpected outcome of our investigation is that $\lambda\mu$'s contextual reduction and substitution are naturally representable in the π-calculus, and are completely dealt with by the encoding, so do not require additional synchronisations.

[1] The notion of lazy reduction has different definitions in the literature, depending on the object language of choice. Here that notion corresponds to 'call by name reduction'; to stay within the terminology of the original papers, we will use the word 'Lazy' here.

[2] The name $\Lambda\mu$ was first introduced in [53], that also introduced a different notation for terms, in placing names *behind* terms, rather than in front, as done by Parigot and de Groote; we use their notation here.

From past papers it might seem that the interpretation of 'classical' calculi like $\lambda\mu$ comes at a great expense; for example, to encode *typed* $\lambda\mu$, in [36] an extension of Milner's encoding is defined that uses a π-calculus that is strongly typed; since reduction in $\mathcal{X}$ is not confluent, [10] shows preservation of reduction in $\mathcal{X}$ only with respect to the contextual ordering '$\sqsubseteq_{\mathrm{c}}$' (so not with respect to contextual equivalence '$\sim_{\mathrm{c}}$', nor with respect to weak bisimilarity '$\approx$'); [25] defines a non-compositional interpretation of $\lambda\mu\tilde{\mu}$ that strongly depends on recursion, and does not regard the logical aspect at all.

Our contribution to this line of research is to study an *output-based* encoding of $\lambda\mu$ into the π-calculus; it is an extension of the one we defined for the λ-calculus [12] and is a natural variant of that for $\Lambda\mu$ we presented in [13]; our approach was compared to the traditional input-based one in [34]. In [12, 13], we have shown that those encodings respect *single-step explicit head-reduction* '$\rightarrow_{\mathrm{xH}}$' (a variant of reduction with explicit substitution '$\rightarrow_{\mathrm{x}}$' that only ever replaces the head variable of a term, see Def. 5.1) modulo contextual equivalence '$\sim_{\mathrm{c}}$'; here we restate those properties with respect to weak bisimilarity '$\approx$'. We show that extending the output-based interpretation $\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{l}}\, a$ of λ-terms [12] (where M is a λ-term and a is the name given to its anonymous output) to $\lambda\mu$, adding cases for context binding and naming, gives a very natural interpretation of $\lambda\mu$-terms to processes. In fact, naming and μ-abstraction can be soundly treated separately, so it is perfectly possible to encode $\Lambda\mu$ and our first results in this direction were indeed on that calculus [13]; as we will argue below, to achieve full abstraction here we have to focus on $\lambda\mu$; otherwise we can not always distinguish between looping and inactive computations.

To accurately define the notion of reduction that is modelled by our interpretation, following [12], in [13] we defined (untyped) $\Lambda\mu\mathbf{x}$, a version with explicit substitution [1, 20] of the $\Lambda\mu$-calculus, together with a notion of *explicit head-reduction* '$\rightarrow_{\mathrm{xH}}$', which can be seen as the minimal system (with explicit substitution) to reduce a term to head-normal form, if possible. The advantage of considering explicit substitution rather than the standard implicit substitution as considered in [43, 52] has been strongly argued by us in [12, 13], and makes an important contribution here as well. To better express the relation between '$\rightarrow_{\mathrm{xH}}$' and '$\rightarrow_{\pi}$', here we follow the approach of [12] when defining explicit head reduction for $\lambda\mu$, rather than that of [13]. In [12] we showed that communication in the π-calculus has a fine semantic level of granularity that 'faithfully mimics' explicit substitution, and not the implicit one; we stress this point again with the results presented in this paper, and the relative ease with which these are achieved. In particular, we will show that, for closed λ-terms, our interpretation models '$\rightarrow_{\mathrm{xH}}$' reductions through '$\rightarrow_{\pi}$' synchronisations directly, but modulo garbage collection, *i.e.* removing sub-processes that can no longer interact with others; we will argue that this result is similar to Milner's first result, shown in [43] (see Thm 3.3).

As was the case for Milner's interpretation, our interpretation places sub-terms (in particular, those that are to be substituted, and therefore also the operand in an application) under guarded replication. Since in the pure π-calculus it is not possible to simulate reductions that take place in terms that are placed under guard, the calculus that can be effectively represented is limited (the restriction of not allowing reduction under guard is dropped in [36]); also other interpretations defined in the past do not model full reduction for the same reason. In our case, as in [12], thanks to the fact that abstraction is encoded through an asynchronous output, the restriction is to that of head reduction.

Although the notion of structural reduction in $\lambda\mu$ is very different from normal β-reduction, no special measures had to be taken in order to be able to express it through our interpretation. The component of the interpretation that deals with pure λ-terms is almost exactly that of [12] (ignoring for the moment that substitution is modelled using a guard, which affects also the interpretation of variables), but for the use of replication in the case for application. In fact, the distributive character of application in $\lambda\mu$, and of both term and context substitution is

dealt with entirely by structural congruence in π (see also Example 6.9), and both naming and context binding are dealt with statically, by the interpretation. In fact, through our encoding it becomes clear that explicit structural substitution is just a distributed variant of application (see Remark 6.5).

We will show a number of results that in part we also showed in [12, 13] for the λ and $\Lambda\mu$ calculi, respectively. In Thm 7.1, we show that single-step explicit head reduction is respected by the encoding in such a way that each β-reduction step is implemented through at least one synchronisation; this leads to operational soundness and completeness results. In Thm 7.6 we show that the encoding also respects equality on $\lambda\mu\mathbf{x}$, but modulo weak bisimulation, and in Thm 7.7 that it gives a semantics for $\lambda\mu$. Since our encoding deals with head reduction as well as open terms, an operation of renaming is needed that is part of weak bisimilarity; in Thm 9.2 we will show that to model lazy reduction on closed terms, this renaming is not needed.

We will show that all results we show for $\lambda\mu$ can be shown for the λ-calculus as well; in Thm A.7 we show that to model lazy reduction on closed λ-terms, the only part of weak bisimilarity that is needed is garbage collection.

Full abstraction is an important property stated for semantics of programming languages and formal calculi. Given a semantics, which interprets terms of a source language or calculus into a domain or target language, full abstraction expresses that all interpretations of terms that are equal in the target domain are equal also under reduction, or operational semantics, of the source. This property is not alway easily achieved. For example, for the standard λ-calculus, the interpretation of terms through Böhm trees [15] gives a semantics that is not fully abstract with respect to the notion of β-equality, since terms that are not related through reduction can have the same Böhm tree. Similarly, for models created using the intersection type discipline [16, 7], terms that can be assigned the same sets of types need not be related by reduction. Moreover, it can be that in the target language operations are permitted that do no correspond to operations of the interpreted language.

When interpreting λ-terms into the π-calculus, an abstraction $\lambda x.M$ has to be mapped to a process willing to interact with its context (which would be the interpretation of the applicative context in which the abstraction occurs). Since this process can interact, in particular it will have a channel name a over which this interaction can take place immediately, be it through input or output over a, which means that the process contains at least an unguarded input or output. Bisimilarity of processes is typically defined over the capacity of processes to interact, to produce an input or output, and thereby the interpretation of an abstraction *has to be* a process that is not bisimilar to the (inactive) process $\mathtt{0}$. To achieve full abstraction, terms that are incapable of interacting with their context, like $\Delta\Delta$ (with $\Delta = \lambda x.xx$), cannot be mapped unto a process that allows interaction, so should be mapped to a process bisimilar to $\mathtt{0}$. This then immediately implies that the interpretations of $\lambda y.\Delta\Delta$ and $\Delta\Delta$ *cannot be* bisimilar. However, these terms are both *unsolvable*, have both no head-normal form, and have the same set of approximants [57, 58] (*i.e.* the same Böhm tree), so are equated under approximation semantics. This then implies that any interpretation of the λ-calculus (or $\lambda\mu$ for that matter) into the π-calculus cannot achieve full abstraction with respect to any standard semantics (based on β-reduction).

However, under *weak* semantics (developed in detail in this paper for $\lambda\mu$), based on lazy reduction [3], the λ-terms $\lambda y.\Delta\Delta$ and $\Delta\Delta$ *are* distinguished. As we will illustrate in this paper, it turns out that this notion is exactly the notion of equality that is respected by any fully abstract interpretation of the λ-calculus into the π-calculus. Sangiorgi was the first to show a full abstraction result [51, 52] for (essentially) Milner's encoding $\llbracket M \rrbracket^{\mathrm{M}} a$, by showing that $\llbracket M \rrbracket^{\mathrm{M}} a \approx \llbracket N \rrbracket^{\mathrm{M}} a$ if and only if $M \simeq N$, where '$\simeq$' is the *applicative bisimilarity* on λ-terms [4].

However, this result comes at a price: applicative bisimulation equates terms that are not weakly bisimilar under the interpretation. To solve this, Sangiorgi extends the encoding to Λ_c, a λ-calculus enriched with constants and changes it into a mapping onto the *Higher Order* π-calculus, a variant of the π-calculus with higher-order communications.

To achieve a full-abstraction result for our interpretation we will use a new and considerably *different* technique: rather than reason through applicative bisimulation, we reason through weak approximation semantics for $\lambda\mu$, which will be defined in this paper. In particular, most of the technical development in this paper is dedicated to defining the correct notions for $\lambda\mu$, and showing the necessary properties (see Sect. 4, 5, 8, and 11).

First we characterise what is exactly the equivalence between terms in $\lambda\mu$ that is representable in the π-calculus through our encoding $\llbracket \cdot \rrbracket^{\mathrm{L}}_{\mathrm{J}} \cdot$; as for Sangiorgi, this turns out to be *weak* equivalence (see Sect. 10), that essentially equates terms that have the same $\lambda\mu$-Lévy-Longo tree [40, 42] (for the pure λ-calculus, those are a lazy variant of Böhm trees), which corresponds to the set of weak approximants; a notable difference between ours and Sangiorgi's result is that we deal with *all terms*, not just the closed ones.

In Thm 7.1 we will show that our interpretation respects '$\to_{\mathrm{xH}}$' modulo '$\approx$', and in Thm 7.6 that it even models '$=_{\mathrm{x}}$', the congruence generated by '$\to_{\mathrm{x}}$', from which a similar result for '$=_{\beta\mu}$' follows directly. In Thm 12.1, we extend this result to *weak explicit head equivalence* '$\sim_{w\mathrm{xH}}$', the equivalence relation generated by '$\to_{\mathrm{xH}}$' that equates also terms without weak head-normal form. The main proof of the full abstraction result is then achieved through showing that '$\sim_{w\mathrm{xH}}$' equates to '$\sim_{w\beta\mu}$', the equivalence relation generated by standard reduction that also equates terms without weak head normal form: this latter result is stated entirely within $\lambda\mu$ and does not depend on the encoding. To achieve this, we define a choice of operational equivalences for the $\lambda\mu$-calculus, both with and without explicit substitution. Next to '$\sim_{w\mathrm{xH}}$' we define *weak head equivalence* '$\sim_{w\mathrm{H}}$' and show that for $\lambda\mu$-terms without explicit substitution, '$\sim_{w\mathrm{xH}}$' corresponds to '$\sim_{w\mathrm{H}}$'. Following essentially [57, 58], we also define a notion of weak approximation and show that the relations '$\sim_{\mathcal{A}_w}$', which expresses that terms have the same set of weak approximants, '$\sim_{w\mathrm{H}}$', and '$\sim_{w\beta\mu}$' all correspond. The combination of these results then yields full abstraction.

Of course the full abstraction result is achievable for the pure λ-calculus as well; although this cannot simply follow from the results we will show below for $\lambda\mu$, the proofs needed are almost carbon copies, removing all treatment of context switches. The interpretation of terms into the π-calculus is slightly easier, and a more direct relation between explicit head reduction and synchronisation can be established. The treatment of explicit head reduction also facilitates a reformulation of Milner's first result, which show a direct, step-by-step relation between weak head reduction for the λ-calculus (also known as lazy reduction) and synchronisation inside the image of terms under Milner's interpretation.

Organisation of this paper We start with revisiting the $\lambda\mu$-calculus in Sect. 1 and define a notion of *head-reduction* '$\to_{\mathrm{H}}$'. In Sect. 2 we revisit the π-calculus, enriched with *pairing*, and will discuss some of the historic context and background of our work in Sect. 3. In Sect. 4 we define $\lambda\mu\mathbf{x}$, a version of $\lambda\mu$ with *explicit substitution*, as well as a notion of *explicit head-reduction* in Sect. 5, and in Sect. 6 define our *logical interpretation* of $\lambda\mu\mathbf{x}$ in to π and prove a soundness result for explicit head-reduction with respect to weak bisimilarity in Sect. 7. In Sect. 9 we will show that the operation of renaming we have defined in Sect. 7 is not needed when dealing weak (*i.e.* lazy) reduction on closed terms, so is the price to pay when modelling head reduction and on open terms.

Working towards our full abstraction result, *i.e* that $\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}} a \approx \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}} a$ if and only if $M \sim_{w\beta\mu} N$, in Sect. 8 we will define notions of weak reduction, in particular *weak head reduction* and *weak explicit head reduction*. In Section 10 we define the two notions of equivalence these induce,

respectively '$\sim_{w\text{H}}$' and '$\sim_{w\text{XH}}$', also equating terms without weak head-normal form and show that these notions coincide on pure $\lambda\mu$ terms (*i.e.* without explicit substitutions). We also define the equivalence '$\sim_{w\beta\mu}$' induced by '$\rightarrow_{\beta\mu}$' on pure $\lambda\mu$ terms, that also equates terms without weak head-normal form. In Sect. 11, we define a notion of *weak approximation* for $\lambda\mu$, and show the semantics this induces, which corresponds to Lévy-Longo trees, is fully abstract with respect to both '$\sim_{w\text{H}}$' and '$\sim_{w\beta\mu}$'.

Then, in Sect. 12, we will show that our logical interpretation is fully abstract with respect to weak bisimilarity '$\approx$' on processes and the equivalences '$\sim_{w\text{XH}}$', '$\sim_{w\text{H}}$', '$\sim_{\mathcal{A}_w}$', and '$\sim_{w\beta\mu}$' on pure $\lambda\mu$-terms. To conclude, in Sect. A, we will focus on the λ-calculus, and state the results that are provable when removing context switches; an interesting one is the reformulation of Milner's result (Thm 3.3), but now with explicit weak head reduction.

This paper is an extended and improved version of [13, 14], but dealing with $\lambda\mu$, rather than $\Lambda\mu$ as in [13].

Notation We will write $\underline{n}$ for the set $\{1,\ldots,n\}$. We will use a vector notation $\vec{\cdot}$ as abbreviation for any sequence: for example, $\vec{x_i}$ stands for $x_1,\ldots,x_n$, for any irrelevant n, or for $\{x_1,\ldots,x_n\}$, and $\langle\overrightarrow{\alpha_i := N_i \cdot \beta_i}\rangle$ for $\langle\alpha_1 := N_1 \cdot \beta_1\rangle \cdots \langle\alpha_n := N_n \cdot \beta_n\rangle$, etc. When possible, we will drop the indices. We also use '$\triangleq$' rather than '$\Longleftrightarrow$' for the symbol representing 'is defined as'; the latter represents a logical implication, which we feel would be misplaced.

1 The $\lambda\mu$ calculus

In this section, we will briefly discuss Parigot's $\lambda\mu$-calculus [48]; we assume the reader to be familiar with the λ-calculus and its notion of reduction '$\rightarrow_\beta$' and equality '$=_\beta$', so will be brief on details. In the next section we will define explicit head-reduction for $\lambda\mu\mathbf{x}$, a variant of $\lambda\mu$ with explicit substitution *à la* $\lambda\mathbf{x}$ [20], and will show full abstraction results for $\lambda\mu\mathbf{x}$; since $\lambda\mu\mathbf{x}$ implements $\lambda\mu$-reduction, this implies that, automatically, our main results are also shown for standard reduction (with implicit substitution).

$\lambda\mu$ is a proof-term syntax for classical logic expressed in Natural Deduction, defined as an extension of the Curry type assignment system for the λ-calculus by adding *naming* $[\alpha]M$ and *context binding* $\mu\alpha.M$ features, as well as *structural reduction* (see Def. 1.4). In $\lambda\mu$, the naming and context binding features always come together as in $\mu\alpha.[\beta]M$; in $\Lambda\mu$, they can be used separately, so there also $\mu\alpha.[\lambda x.x]$is a term, albeit untypeable. The naming feature $[\alpha]M$ expresses that α serves as label for the term M, and $\mu\alpha.M$ is used to redirect operands (terms) to those labeled α inside M. A *context switch* $\mu\alpha.[\beta]M$ now expresses that the focus of the derivation (proof), to which the term corresponds, changes; the idea is that the applicative context of M is not meant for that term itself, but rather for its sub-terms labeled with α. It is the naming feature, together with the structural rules, that make $\lambda\mu$ difficult to reason over; this is reflected in [33] and [11], where the interpretation of $\lambda\mu$ into $\lambda\mu\tilde{\mu}$ [26] and $\mathcal{X}$ (as introduced in [11]), respectively, does not respect reduction.

Definition 1.1 (Syntax of $\lambda\mu$) The *$\lambda\mu$-terms* we consider are defined over the set of *variables* represented by Roman characters, and *names*, or *context* variables, represented by Greek characters, through the grammar:

$$\begin{array}{rcll} M,N & ::= & x & (\textit{variable}) \\ & | & \lambda x.M & (\textit{abstraction}) \\ & | & MN & (\textit{application}) \\ & | & \mu\alpha.[\beta]M & (\textit{context switch}) \end{array}$$

We will occasionally write C for the pseudo-term $[\alpha]M$, and use $\lambda\mu$ also for the set of all $\lambda\mu$-terms.

The main difference between $\Lambda\mu$ and $\lambda\mu$ is that in the former, $[\alpha]M$ is considered to be a term.

As usual, $\lambda x.M$ binds x in M, and $\mu\alpha.\mathrm{C}$ binds α in C, and the notions of free variables $fv(M)$ and names $fn(M)$ are defined accordingly; the notion of α-conversion extends naturally to bound names and we assume Barendregt's convention in that we assume that free and bound variables and names are always distinct, using α-conversion when necessary. As usual, we call a term *closed* if it has no free variables or names.

Denotational semantics of $\lambda\mu$ has been studied by Streicher and Reus [55], who presented a domain theoretic model of $\lambda\mu$ using a *model of continuations*. They proposed a model of both typed and untyped λ-calculi embodying a concept of continuation, including Felleisen's $\lambda\mathcal{C}$-calculus [29, 28] and a version of Parigot's $\lambda\mu$. Their model is based on the solution of the domain equations $D = C \to R$ and $C = D \times C$, where R is an arbitrary domain of 'results'. The domain C is the set of what are called 'continuations' in [55], which are *streams*, infinite tuples of elements in D; D is the domain of continuous functions from C to R and is the set of 'denotations' of terms.

Simple type assignment for $\lambda\mu$ is defined as follows:

Definition 1.2 (Types, Contexts, and Typing) *i*) Types are defined by the grammar:

$$A, B \ ::= \ \varphi \mid A \to B$$

where φ is a basic type of which there are countably many.

ii) A *context of inputs* Γ is a mapping from term variables to types, denoted as a finite set of *statements* $x{:}A$, such that the *subject* of the statements (x) are distinct. We write Γ_1, Γ_2 for the *compatible* union of Γ_1 and Γ_2 (if $x{:}A_1 \in \Gamma_1$ and $x{:}A_2 \in \Gamma_2$, then $A_1 = A_2$), and write $\Gamma, x{:}A$ for $\Gamma, \{x{:}A\}$.

iii) Contexts of *outputs* Δ as mappings from names to types, and the notions Δ_1, Δ_2 and $\alpha{:}A, \Delta$ are defined similarly.

iv) Type assignment for $\lambda\mu$ is defined by the following natural deduction system.

$$(Ax): \ \frac{}{\Gamma, x{:}A \vdash x : A \mid \Delta} \qquad (\mu): \ \frac{\Gamma \vdash M : B \mid \alpha{:}A, \Delta}{\Gamma \vdash \mu\alpha.[\beta]M : A \mid \beta{:}B, \Delta}\ (\alpha \notin \Delta) \quad \frac{\Gamma \vdash M : A \mid \alpha{:}A, \Delta}{\Gamma \vdash \mu\alpha.[\alpha]M : A \mid \Delta}\ (\alpha \notin \Delta)$$

$$(\to I): \ \frac{\Gamma, x{:}A \vdash M : B \mid \Delta}{\Gamma \vdash \lambda x.M : A \to B \mid \Delta}\ (x \notin \Gamma) \qquad (\to E): \ \frac{\Gamma \vdash M : A \to B \mid \Delta \quad \Gamma \vdash N : A \mid \Delta}{\Gamma \vdash MN : B \mid \Delta}$$

We write $\Gamma \vdash_{\lambda\mu} M : A \mid \Delta$ for derivable judgements in this system.

So, for the context $\Gamma, x{:}A$, we have either $x{:}A \in \Gamma$, or Γ is not defined on x; notice that in the first variant of rule (μ), $\beta{:}B$ is added to Δ; β can already appear in Δ, but then has to have the same type; on the other hand, that rule removes $\alpha{:}A$ from the right context.

In [9], together with Barbanera and de'Liguoro, the first author extracted an intersection type syntax and the corresponding type theory out of the construction of Streicher and Reus's model, and showed that this yields a *filter* model for $\lambda\mu$. This was followed by [8] where the first author studied a version of the system of [9] with *strict* types, for which the characterisation of various notions of termination is shown.

In $\lambda\mu$, reduction of terms is expressed via implicit substitution; as usual, $M\{N/x\}$ stands for the substitution of all occurrences of x in M by N, and $M\{N{\cdot}\gamma/\alpha\}$, the *structural substitution*, for the term obtained from M when every (pseudo) sub-term of the form $[\alpha]M'$ is replaced by $[\gamma]M'N$.[3] For reasons of clarity, and because below we will present a version of $\lambda\mu$ that makes the substitution explicit, we define the structural substitution formally.

[3] This notion is often defined as $M\{N/\alpha\}$, where every (pseudo) sub-term of the form $[\alpha]M'$ is replaced by

Definition 1.3 (Structural substitution) We define $M\{N{\cdot}\gamma/\alpha\}$ (where γ is fresh, α does not occur bound in M, and every sub-term $[\alpha]L$ of M is replaced by $[\gamma]LN$) by induction over the structure of (pseudo-)terms by:

$$\begin{array}{rcl@{\qquad}rcl}
([\alpha]M)\,\{N{\cdot}\gamma/\alpha\} & \triangleq & [\gamma](M\,\{N{\cdot}\gamma/\alpha\})N & x\,\{N{\cdot}\gamma/\alpha\} & \triangleq & x \\
([\beta]M)\,\{N{\cdot}\gamma/\alpha\} & \triangleq & [\beta](M\,\{N{\cdot}\gamma/\alpha\})\ (\alpha\neq\beta) & (\lambda x.M)\,\{N{\cdot}\gamma/\alpha\} & \triangleq & \lambda x.M\,\{N{\cdot}\gamma/\alpha\} \\
(\mu\beta.\mathrm{C})\,\{N{\cdot}\gamma/\alpha\} & \triangleq & \mu\beta.\mathrm{C}\,\{N{\cdot}\gamma/\alpha\} & (PQ)\,\{N{\cdot}\gamma/\alpha\} & \triangleq & P\,\{N{\cdot}\gamma/\alpha\}\ Q\,\{N{\cdot}\gamma/\alpha\}
\end{array}$$

We have the following rules of computation in $\lambda\mu$:

Definition 1.4 ($\lambda\mu$ reduction) *i*) $\lambda\mu$ has a number of reduction rules: two *computational rules*

$$\begin{array}{rrcl}
\textit{logical}\ (\beta): & (\lambda x.M)N & \to & M\{N/x\} \\
\textit{structural}\ (\mu): & (\mu\alpha.\mathrm{C})N & \to & \mu\gamma.(\mathrm{C}\{N{\cdot}\gamma/\alpha\})
\end{array}$$

as well as the *simplification rules*

$$\begin{array}{rrcl}
\textit{renaming}: & \mu\alpha.[\beta]\mu\gamma.[\delta]M & \to & \begin{cases}\mu\alpha.[\beta]M\{\beta/\gamma\} & (\delta=\gamma)\\ \mu\alpha.[\delta]M\{\beta/\gamma\} & (\delta\neq\gamma)\end{cases} \\
\textit{erasing}: & \mu\alpha.[\alpha]M & \to & M\ \ (\alpha\notin fn(M))
\end{array}$$

which are added mainly to simplify the presentation of results.

ii) We use the contextual rules:[4]

$$M\to N\ \Rightarrow\ \begin{cases} ML \to NL \\ LM \to LN \\ \lambda x.M \to \lambda x.N \\ \mu\alpha.[\beta]M \to \mu\alpha.[\beta]N \end{cases}$$

iii) We use '$\to^*_{\beta\mu}$' for the pre-congruence[5] based on these rules, '$=_{\beta\mu}$' for the congruence, write $M\to^{nf}_{\beta\mu}N$ if $M\to^*_{\beta\mu}N$ and N is in normal form, $M\to^{hnf}_{\beta\mu}N$ if $M\to^*_{\beta\mu}N$ and N is in head-normal form, $M{\Downarrow}$ if there exists a finite reduction path starting from M,[6] and $M{\Uparrow}$ if this is not the case.

We will use these notations for other notions of reduction as well, sometimes subscripted for clarity.

That this notion of reduction is confluent was shown in [50]; so we have:

Proposition 1.5 If $M=_{\beta\mu}N$ and $M\to^*_{\beta\mu}P$, then there exists Q such that $P\to^*_{\beta\mu}Q$ and $N\to^*_{\beta\mu}Q$.

The intuition behind the structural rule is given by [32]: *'in a $\lambda\mu$-term $\mu\alpha.M$ of type $A{\to}B$, only the subterms named by α are* really *of type $A{\to}B$ (...); hence, when such a μ-abstraction is applied to an argument, this argument must be passed over to the sub-terms named by α."* We can think of $[\alpha]M$ as storing the type of M amongst the alternative conclusions by naming it α.

Parigot showed in [49] that typeable terms are strongly normalisable. That paper also defines the extensional rules

$$\begin{array}{rrcll}
(\eta): & \lambda x.Mx & \to & M & (x\notin fv(M)) \\
(\eta\mu): & \mu\alpha.[\beta]M & \to & \lambda x.\mu\gamma.[\beta]M\{x{\cdot}\gamma/\alpha\} &
\end{array}$$

$[\alpha]M'N$; in our opinion, this creates confusion, since α gets 'reintroduced'; it is in fact a new name, which is illustrated by the fact that, in a system with types, α then changes type, as also expressed by rule (*C-sub*) in Def. 4.4. Moreover, when making this substitution explicit, bound and free occurrences of the same name would be introduced, violating Barendregt's convention.

[4] Normally the contextual rules are not mentioned but are left implicit; we state them here, since we will below consider notions of reduction that do not permit all contextual rules.

[5] A *pre-congruence* is a reflexive and transitive relation that is preserved in all contexts; a *congruence* is symmetric pre-congruence.

[6] Note that this does not imply that *all* paths are finite.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x{:}(A\to B)\to A \vdash x : (A\to B)\to A \mid \alpha{:}A}\,(Ax) \quad \dfrac{\dfrac{\dfrac{}{x{:}(A\to B)\to A, y{:}A \vdash y : A \mid \alpha{:}A, \beta{:}B}\,(Ax)}{x{:}(A\to B)\to A, y{:}A \vdash \mu\beta.[\alpha]y : B \mid \alpha{:}A}\,(\mu)}{x{:}(A\to B)\to A \vdash \lambda y.\mu\beta.[\alpha]y : A\to B \mid \alpha{:}A}\,(\to I)}{x{:}(A\to B)\to A \vdash x\,(\lambda y.\mu\beta.[\alpha]y) : A \mid \alpha{:}A}\,(\to E)}{x{:}(A\to B)\to A \vdash \mu\alpha.[\alpha]\,(x\,(\lambda y.\mu\beta.[\alpha]y)) : A \mid}\,(\mu)}{\vdash \lambda x.\mu\alpha.[\alpha]\,(x\,(\lambda y.\mu\beta.[\alpha]y)) : ((A\to B)\to A)\to A \mid}\,(\to I)
$$

Figure 1: A derivation for a term representing Peirce's Law in $\vdash_{\lambda\mu}$

We do not consider these rules here: the model we present through our interpretation is not extensional, and we can therefore not show that those rules are preserved by the interpretation.

Example 1.6 As an example illustrating the fact that this system is more powerful than the system for the λ-calculus, Fig. 1 shows that it is possible to inhabit Peirce's Law (due to [47]). The underlying logic of the system of Def. 1.2 corresponds to *minimal classical logic* [5].

We also consider the notion of *head reduction*; it is defined in [57] for the λ-calculus by first defining the head-redex of a term as the subterm $(\lambda y.M)\,N$ in a term of the form

$$\lambda x_1 x_2 \cdots x_n.(\lambda y.M)\,N L_1 L_2 \cdots L_m \quad (n \geq 0, m \geq 0)$$

Head reduction is then that notion in which each step is determined by contraction of the *head redex* only (when it exists); head-normal forms (the normal forms with respect to head reduction) are of the generic shape

$$\lambda x_1 x_2 \cdots x_n.y L_1 L_2 \cdots L_m \quad (n \geq 0, m \geq 0)$$

and y in this term is called the *head variable*. In $\lambda\mu$, given the naming and μ-binding features, the notion of head redex is not this easily defined; rather, here we define head reduction by not allowing reductions to take place in the right-hand side of applications (in the context of the λ-calculus, this gives the original notion); we also define a notion of head-normal form for $\lambda\mu$.

Definition 1.7 (Head reduction for $\lambda\mu$ (*cf.* [39])) *i*) We define *head reduction* '$\to_{\mathrm{H}}$' as the restriction of '$\to_{\beta\mu}$' by removing the contextual rule:

$$M \to N \;\Rightarrow\; LM \to LN$$

ii) The $\lambda\mu$ *head-normal forms* (hnf) are defined through the grammar:

$$
\begin{array}{rcll}
\boldsymbol{H} & ::= & \lambda x.\boldsymbol{H} & \\
 & \mid & x M_1 \cdots M_n & (n \geq 0) \\
 & \mid & \mu\alpha.[\beta]\boldsymbol{H} & (\beta \neq \alpha \text{ or } \alpha \in \boldsymbol{H}, \text{ and } \boldsymbol{H} \neq \mu\gamma.[\delta]\boldsymbol{H}')
\end{array}
$$

Notice that the $\to_{\beta\mu}$-hnfs are $\to_{\mathrm{H}}$-normal forms.

The following is straightforward:

Proposition 1.8 ('$\to_{\mathrm{H}}$' implements $\lambda\mu$'s head reduction) *If $M \to^*_{\beta\mu} N$ with N in* hnf *(so $M \to^{hnf}_{\beta\mu} N$), then there exists $\boldsymbol{H}$ such that $M \to^{nf}_{\mathrm{H}} \boldsymbol{H}$ (so $\boldsymbol{H}$ is in $\to_{\mathrm{H}}$-normal form) and $\boldsymbol{H} \to^*_{\beta\mu} N$ without using '$\to_{\mathrm{H}}$'.*

Notice that $\lambda f.(\lambda x.f(xx))\,(\lambda x.f(xx)) \;\to_{\mathrm{H}}\; \lambda f.f\,((\lambda x.f(xx))\,(\lambda x.f(xx)))$ and that this last term is in hnf, and in $\to_{\mathrm{H}}$-normal form.

2 The synchronous π-calculus with pairing

The notion of π-calculus that we consider in this paper was already considered in [12] and is different from other systems studied in the literature [35] in that it adds *pairing* and uses a *let*-construct to deal with inputs of pairs of names that get distributed, similar to that defined in [2]; in contrast to [10, 12], we do not consider the asynchronous π-calculus.

As already argued in [12], the main reason for the addition of pairing lies in preservation of (implicate, or functional) type assignment;[7] therefore *data* is introduced as a structure over names, such that not only names but also pairs of names can be sent (but not a pair of pairs); this way a channel may pass along either a name or a pair of names.

Definition 2.1 (Processes) *i*) *Channel names* and *data* are defined by:

$$a,b,c,d,x,y,z \quad \textit{names} \qquad\qquad p \;::=\; a \mid \langle a,b\rangle \quad \textit{data}$$

ii) Processes are defined by:

$$\begin{array}{rcllll} P,Q & ::= & 0 & (\textit{nil}) & \mid\; a(x).P & (\textit{input}) \\ & \mid & P \mid Q & (\textit{composition}) & \mid\; \overline{a}\,p.P & (\textit{output}) \\ & \mid & !P & (\textit{replication}) & \mid\; \mathit{let}\,\langle x,y\rangle = p \;\mathit{in}\; P & (\textit{let construct}) \\ & \mid & (\nu a)\,P & (\textit{restriction}) & & \end{array}$$

iii) We see, as usual, ν as a binder, and call the name n *bound* in $(\nu n)\,P$, x bound in $a(x).P$ and x,y bound in $\mathit{let}\,\langle x,y\rangle = p \;\mathit{in}\; P$; we write $bn(P)$ for the set of bound names in P; n is *free* in P if it occurs in P but is not bound, and we write $fn(P)$ for the set of free names in P. We call a in $(\nu a)\,P$ a *hidden* channel.

iv) A *context* $\mathsf{C}\lceil\cdot\rfloor$ is a process with a single hole, and we write $\mathsf{C}\lceil P\rfloor$ for filling the hole with P.

v) We call $a(x)$ and $\overline{a}\,p$ *guards*, and call P in $a(x).P$ and $\overline{a}\,p.P$ a process *under guard*.

vi) We will abbreviate $a(x).\mathit{let}\,\langle y,z\rangle = x \;\mathit{in}\; P$ by $a(y,z).P$, as well as $(\nu m)\,(\nu n)\,P$ by $(\nu mn)\,P$, and write $\overline{a}\,p$ for $\overline{a}\,p.0$.

vii) As in [52], we write $a{\rightarrow}\overline{b}$ for the *forwarder* $a(x).\overline{b}\,x$, and $\overline{x}(w).P$ for $(\nu w)\,(\overline{x}\,w.P)$.

Notice that the pairing in data is *not* recursive. Data occurs only in two cases: $\mathit{let}\,\langle x,y\rangle = p \;\mathit{in}\; P$ and $\overline{a}\,p$, and then p is either a single name, or a pair of names; we therefore do not allow $a(\langle x,y\rangle).P$, nor $\overline{a}\,\langle\langle b,c\rangle,d\rangle.P$, nor $\overline{\langle b,c\rangle}\,p.P$, nor $(\nu\langle a,b\rangle)\,P$, nor $\mathit{let}\,\langle\langle a,b\rangle,y\rangle = p \;\mathit{in}\; P$, etc. So substitution $P\{p/x\}$ is a partial operation, which depends on the places in P where x occurs; whenever we use $P\{p/x\}$, we will assume it is well defined. It is worthwhile to point out that using pairing is not the same as working with the polyadic (or even dyadic) π-calculus, because there each channel has a fixed arity, whereas we allow data to be sent, which is either a name or a pair of names.

Definition 2.2 (Structural Congruence) The *structural congruence* is the smallest congruence generated by the rules:

$$\begin{array}{rcl@{\qquad}rcll} P \mid 0 & \equiv & P & (P \mid Q) \mid R & \equiv & P \mid (Q \mid R) & \\ P \mid Q & \equiv & Q \mid P & (\nu m)\,(\nu n)\,P & \equiv & (\nu n)\,(\nu m)\,P & \\ !P & \equiv & P \mid !P & (\nu n)\,(P \mid Q) & \equiv & P \mid (\nu n)\,Q & (n \notin fn(P)) \\ (\nu n)\,0 & \equiv & 0 & \mathit{let}\,\langle x,y\rangle = \langle a,b\rangle \;\mathit{in}\; P & \equiv & P\{a/x,b/y\} & \end{array}$$

As usual, we will consider processes modulo congruence and α-conversion: this implies that we will not deal explicitly with the process $\mathit{let}\,\langle x,y\rangle = \langle a,b\rangle \;\mathit{in}\; P$, but rather with $P\{a/x,b/y\}$.

[7] If we would not consider types in this paper, we could consider the standard π-calculus; however, the details of the interpretation would change (more replication would be needed).

Because parallel composition is associative, we will normally not write brackets in a parallel composition of more than two processes.

Computation in the π-calculus with pairing is expressed via the exchange of *data*.

Definition 2.3 (Reduction) *i*) The *reduction relation* over the processes of the π-calculus with pairing is defined by the following (elementary) rules:

$$\begin{array}{rcll} \overline{a}\,p.P \mid a(x).Q & \rightarrow_\pi & P \mid Q\{p/x\} & (\textit{synchronisation}) \\ P \rightarrow_\pi P' & \Rightarrow & (\nu n)\,P \rightarrow_\pi (\nu n)\,P & (\textit{hiding}) \\ P \rightarrow_\pi P' & \Rightarrow & P \mid Q \rightarrow_\pi P' \mid Q & (\textit{composition}) \\ P \equiv Q \wedge Q \rightarrow_\pi Q' \wedge Q' \equiv P' & \Rightarrow & P \rightarrow_\pi P' & (\textit{structural congruence}) \end{array}$$

We write $P \rightarrow_\pi (c)\ Q$ if P reduces to Q in a single step via a synchronisation over channel c, and write '$\rightarrow_\pi (=_\alpha)$' if we want to point out that α-conversion has taken place during the synchronisation. We say that $P \rightarrow_\pi (c)\ Q$ takes place *over a hidden channel* if c is hidden in P.

ii) We say that a P is *irreducible* (is in *normal form*) if it does not contain a possible synchronisation, *i.e.* P is not of the shape $(\nu\vec{b})\,(\overline{a}\,p.Q \mid a(x).R \mid S)$ (up to structural congruence).

Notice that $\mathit{let}\,\langle x,y\rangle = a\ \mathit{in}\ P$ (where a is a name) is *stuck*. Also,

$$\begin{array}{rcl} \overline{a}\langle b,c\rangle \mid a(x,y).Q & \triangleq & \overline{a}\,\langle b,c\rangle \mid a(z).\mathit{let}\,\langle x,y\rangle = z\ \mathit{in}\ Q \\ & \rightarrow_\pi & \mathit{let}\,\langle x,y\rangle = \langle b,c\rangle\ \mathit{in}\ Q \\ & \equiv & Q\{b/x,c/y\} \end{array}$$

There are several notions of equivalence defined for the π-calculus: the one we consider here, and will show is related to our encoding, is that of weak bisimilarity.

Definition 2.4 (Weak bisimilarity) *i*) We write $P{\downarrow}\overline{n}$ and say that P *outputs on* n (or P exhibits an output barb on n) if $P \equiv (\nu\vec{b})\,(\overline{n}\,p.Q \mid R)$, where $n \notin \vec{b}$ and $P{\downarrow}n$ (P *inputs on* n) if $P \equiv (\nu\vec{b})\,(n(x).Q \mid R)$, where $n \notin \vec{b}$.

ii) We write $P{\Downarrow}\overline{n}$ (P *will output on* n) if there exists Q such that $P \rightarrow_\pi^* Q$ and $Q{\downarrow}\overline{n}$, and $P{\not\Downarrow}_o$ if there exists no n such that $P{\Downarrow}\overline{n}$ (P will not output). Likewise, we write $P{\Downarrow}n$ (P *will input on* n) if there exists Q such that $P \rightarrow_\pi^* Q$ and $Q{\downarrow}n$, and $P{\not\Downarrow}_i$ if there exists no n such that $P{\Downarrow}n$ (P will not input).

iii) A *barbed bisimilarity* '$\approx$' is the largest symmetric relation such that $P \approx Q$ satisfies the following clauses:

- for every name n: if $P{\downarrow}\overline{n}$ then $Q{\Downarrow}\overline{n}$, and if $P{\downarrow}n$ then $Q{\Downarrow}n$;
- for all P', if $P \rightarrow_\pi^* P'$, then there exists Q' such that $Q \rightarrow_\pi^* Q'$ and $P' \approx Q'$;

iv) *Weak bisimilarity* is the largest symmetric relation on processes '$\approx$' defined by: $P \approx Q$ if and only if $\mathsf{C}\lceil P\rfloor \approx \mathsf{C}\lceil Q\rfloor$ for any context $\mathsf{C}\lceil\cdot\rfloor$.

The following property is needed in the proof of Thm 7.1 and 7.6.

Lemma 2.5 (Private resources lemma (*cf.* [44, 52])) *Let $x \neq c$ at most only be used as output channel in P and Q, and not appear in R, then*

$$\begin{array}{rcll} (\nu x)\,(P \mid Q \mid !x(z).R) & \approx & (\nu x)\,(P \mid !x(z).R) \mid (\nu x)\,(Q \mid !x(z).R) & (1) \\ (\nu x)\,(P \mid Q \mid !x(v,d).R) & \approx & (\nu x)\,((\nu y)\,(P\{y/x\} \mid !y(v,d).R) \mid Q \mid !x(v,d).R), \quad (y\ \textit{fresh}) & (2) \\ (\nu x)\,(c(y).P \mid !x(z).R) & \approx & c(y).((\nu x)\,(P \mid !x(z).R)) & (3) \\ (\nu x)\,(!c(v,d).P \mid !x(z).R) & \approx & !c(v,d).((\nu x)\,(P \mid !x(z).R)) & (4) \\ (\nu x)\,(!\overline{c}\,y.P \mid !x(z).R) & \approx & !\overline{c}\,y.((\nu x)\,(P \mid !x(z).R)) & (5) \end{array}$$

Likewise, let $x \neq c$ only be used as input channel in P and Q, and not appear in R, then

$$
\begin{array}{rcll}
(\nu x)\,(P \mid Q \mid !\overline{x}(w).R) & \approx & (\nu x)\,(P \mid !\overline{x}(w).R) \mid (\nu x)\,(Q \mid !\overline{x}(w).R) & (6)\\
(\nu x)\,(P \mid Q \mid !\overline{x}(w).R) & \approx & (\nu x)\,((\nu y)\,(P\{y/x\} \mid !\overline{y}(w).R) \mid Q \mid !\overline{x}(w).R), \quad (y \textit{ fresh}) & (7)\\
(\nu x)\,(!c(v,d).P \mid !\overline{x}(w).R) & \approx & !c(v,d).((\nu x)\,(P \mid !\overline{x}(w).R)) & (8)\\
(\nu x)\,(!\overline{c}y.P \mid !\overline{x}(w).R) & \approx & !\overline{c}y.((\nu x)\,(P \mid !\overline{x}(w).R)) & (9)
\end{array}
$$

Proof: All parts follow easily. Part 1, 4, and 5 are shown in [44, 52] (see Thm 3.5); part 2 follows from part 1, α-conversion, and structural congruence. The proof for the second group is similar to that for the first. □

Part 4, 5, 8, and 9 are part of (extended) structural congruence in [27].

The following is easy to show.

Proposition 2.6 (Synchronisation over hidden channels is unobservable) *Let p, P, Q not contain a, then*

$$
\begin{array}{rcll}
(\nu a)\,(\overline{a}\,p.P \mid a(x).Q) & \approx & P \mid Q\{p/x\} & (10)\\
(\nu a)\,(\overline{a}\,p.P \mid !a(x).Q) & \approx & P \mid Q\{p/x\} & (11)
\end{array}
$$

Proof: For part (10), this follows from the fact that there is only one synchronisation possible in the left-hand process, and before that is activated, no context can interact with either P or Q, since both occur under guard. After the synchronisation over a, that channel name disappears and the process on the right gets created and only then can a context interact. Part (11) follows similarly, using that $(\nu a)\,(!a(x).Q) \approx 0$. □

3 Context and background of this paper

Milner's input-based encoding

In the past, there have been several investigations of interpretation from the λ-calculus into the π-calculus. Research in this direction started by Milner's interpretation $\llbracket\cdot\rrbracket^{\mathrm{M}}\cdot$ of λ-terms [43]; Milner's interpretation is input based, *i.e.* terms are interpreted under an input name, and Milner shows that the interpretation of closed λ-terms respects large-step *lazy* reduction '$\to_{\mathrm{L}}$' [3] to normal form up to substitution (Thm 3.3); this was later generalised to β-equality, but using weak bisimilarity [52]. It is defined by:

Definition 3.1 (Milner's interpretation [43]) Let a not be a λ-variable.

$$
\begin{array}{rcll}
\llbracket x \rrbracket^{\mathrm{M}}\, a & \triangleq & \overline{x}\,a & \\
\llbracket \lambda x.M \rrbracket^{\mathrm{M}}\, a & \triangleq & a(x).a(b).\llbracket M \rrbracket^{\mathrm{M}}\, b & (b \textit{ fresh})\\
\llbracket MN \rrbracket^{\mathrm{M}}\, a & \triangleq & (\nu c)\,(\llbracket M \rrbracket^{\mathrm{M}}\, c \mid (\nu z)\,(\overline{c}z.\overline{c}a.\llbracket z := N \rrbracket^{\mathrm{M}})) & (c,z \textit{ fresh})\\
\llbracket x := M \rrbracket^{\mathrm{M}} & \triangleq & !x(w).\llbracket M \rrbracket^{\mathrm{M}}\, w & (w \textit{ fresh})
\end{array}
$$

Milner calls $\llbracket x := M \rrbracket^{\mathrm{M}}$ an *environment entry*; it could be omitted from the definition above, but is of use separately.

There is a correspondence between Milner's environment entry and a *closure* in Krivine's machine [38], in that a closure is a collection of term substitutions that is created through reduction, as are the environment entries. As in Milner's encoding, term substitutions can only be applied to head variables.

Example 3.2 Using $\llbracket\cdot\rrbracket^{\mathrm{M}}\cdot$, the encoding of a β-redex (only) reduces as follows:

$$
\begin{array}{rcll}
\llbracket (\lambda x.M)N \rrbracket^{\mathrm{M}}\, a & \triangleq & (\nu c)\,(\llbracket \lambda x.M \rrbracket^{\mathrm{M}}\, c \mid (\nu z)\,(\overline{c}z.\overline{c}a.\llbracket z := N \rrbracket^{\mathrm{M}})) & \triangleq\\
 & & (\nu c)\,(c(x).c(b).\llbracket M \rrbracket^{\mathrm{M}}\, b \mid (\nu z)\,(\overline{c}z.\overline{c}a.\llbracket z := N \rrbracket^{\mathrm{M}})) & \to_{\pi}^{+} (c)\\
 & & (\nu z)\,(\llbracket M \rrbracket^{\mathrm{M}}\, \{z/x\}a \mid \llbracket z := N \rrbracket^{\mathrm{M}}) & =_{\alpha} \ (z \notin \llbracket M \rrbracket^{\mathrm{M}}\, a)\\
 & & (\nu x)\,(\llbracket M \rrbracket^{\mathrm{M}}\, a \mid \llbracket x := N \rrbracket^{\mathrm{M}}) & \triangleq\\
 & & (\nu x)\,(\llbracket M \rrbracket^{\mathrm{M}}\, a \mid !x(w).\llbracket N \rrbracket^{\mathrm{M}}\, w) &
\end{array}
$$

Now reduction can continue in (the encoding of) M, but not in N that is guarded by the input on x, which will not be used until the evaluation of $\llbracket M \rrbracket^{\mathrm{M}}_{\mathrm{J}}\, a$ reaches the point where output is generated over x. This implies of course that we can model reductions in M that take place *before* the substitution gets executed, *i.e.* 'under the abstraction', but *after* a first step in the evaluation of the redex: this implies that Milner's encoding represents *more* than just lazy reduction with implicit substitution, and more closely deals with *explicit* substitution; we will make this observation more precise in Thm A.4.

Notice that, in $\llbracket MN \rrbracket^{\mathrm{M}}_{\mathrm{J}}\, a$, the interpretation of the operand N is placed under output (and replication), and thereby blocked from running; this comes at a price: now β-reductions that occur in the right-hand side of an application can no longer be mimicked. Combined with using input to model abstraction, this makes that a redex can only be contracted if it occurs on the outside of a term (is the *top* redex): the modelled reduction is *lazy*, '$\to_{\mathrm{L}}$'.

Milner states an Operational Soundness result for his interpretation:

Theorem 3.3 ([43]) *For closed λ-term M, either $M\Uparrow$ and $\llbracket M \rrbracket^{\mathrm{M}}_{\mathrm{J}}\, u \Uparrow_{\pi}$, or $M \to^*_{\mathrm{L}} \lambda y.R$, and*

$$\llbracket M \rrbracket^{\mathrm{M}}_{\mathrm{J}}\, u \;\to^*_{\pi}\; (\overline{\nu x})\,(\llbracket \lambda y.R \rrbracket^{\mathrm{M}}_{\mathrm{J}}\, u \mid \overrightarrow{\llbracket x := N \rrbracket^{\mathrm{M}}_{\mathrm{J}}}).$$

Although obviously the intention in [43] is that the substitutions $\overrightarrow{\langle x{:=}N\rangle}$ in Thm 3.3 are generated by the reduction (and this is explicitly used in the proof for that result), the way it is formulated this need not necessarily be the case; the result as stated in [43] is therefore not complete. Moreover, it is worthwhile to note that, although not mentioned in [43], the proof of this result treats the substitution as *explicit*, not as *implicit*; for example, in the proof of Lemma 4.5 in that paper, case 3 considers the term $xM_1 \cdots M_n\{N/x\}$ and $NM_1 \cdots M_n\{N/x\}$ to be *different*, whereas in the standard λ-calculus these terms are *identical*. Under explicit substitution, however, the terms $xM_1 \cdots M_n\langle x{:=}N\rangle$ and $NM_1 \cdots M_n\langle x{:=}N\rangle$ do differ, so it is opportune to switch our attention to a calculus with explicit substitution, also when dealing with Milner's interpretation. We will come back to this in Thm A.4, where we restate Milner's result, but formulated with explicit substitution.

For many years, it seemed that Milner had stated the first and final word on the interpretation of the λ-calculus; in fact, input-based interpretations of the λ-calculus into the π-calculus have become the *de facto* standard, and most published systems are based on Milner's interpretation. The various interpretations studied in [52] constitute examples, also in the context of the higher-order π-calculus; [36] used Milner's approach with a typed version of the π-calculus; [56] used it in the context of continuation-passing style languages.

In [44], Milner returned to interpretations of the λ-calculus, but expressed a property over β-reduction, rather than lazy reduction to normal form. To that purpose, he presented a different version of his encoding into the *polyadic* π-calculus. It uses a notion of abstraction $(\lambda a)P$ over processes, but with the restriction that bound names can only be replaced by names (so is not a higher-order feature) and is mainly added for ease of adding definitions. Since only names can be substituted, abstractions can only be applied to names as in $(\lambda a)P\,b$, which stands for (so does not reduce to) $P\{b/a\}$. Also, Milner introduces the notation $\overline{a}.[b_1 \cdots b_n]$, which roughly stands for $\overline{a}\,b_1 \cdots b_n$, and can be used in a synchronisation of the shape

$$v.(\lambda\vec{x})\,P \mid \overline{v}.(\nu\vec{z})\,[\vec{y}]Q \;\to_{\pi}\; (\nu\vec{z})\,(P\{\overrightarrow{y/x}\} \mid Q)$$

provided that $|\vec{x}| = |\vec{y}|$. The new version of Milner's encoding is:

$$\begin{array}{rcll}
\llbracket x \rrbracket^{\mathrm{P}}_{\mathrm{J}} & \triangleq & (\lambda u)\,\overline{x}.[u] & (u \textit{ fresh}) \\
\llbracket \lambda x.M \rrbracket^{\mathrm{P}}_{\mathrm{J}} & \triangleq & (\lambda u)\,u.(\lambda x)\llbracket M \rrbracket^{\mathrm{P}}_{\mathrm{J}} & (u \textit{ fresh}) \\
\llbracket MN \rrbracket^{\mathrm{P}}_{\mathrm{J}} & \triangleq & (\lambda u)\,(\nu v)\,(\llbracket M \rrbracket^{\mathrm{P}}_{\mathrm{J}}\, v \mid (\nu z)\,(\overline{v}.[zu] \mid !z.\llbracket N \rrbracket^{\mathrm{P}}_{\mathrm{J}})) & (u,v,z \textit{ fresh})
\end{array}$$

for which Milner shows that:

Lemma 3.4 ([44]) $(\nu x)\,(\llbracket M \rrbracket^{\mathrm{P}} \mid !x.\llbracket N \rrbracket^{\mathrm{P}}) \approx \llbracket M\{N/x\} \rrbracket^{\mathrm{P}}$

Using Lemma 3.4, Milner shows $\llbracket (\lambda x.M)\,N \rrbracket^{\mathrm{P}} \approx \llbracket M\{N/x\} \rrbracket^{\mathrm{P}}$ (but does not extend this result to $M =_\beta N \Rightarrow \llbracket M \rrbracket^{\mathrm{P}} \approx \llbracket N \rrbracket^{\mathrm{P}}$); see also Thm 7.6 and 7.7 below. As in the proof of Thm 7.6, Milner needs a variant of Lemma 2.5:

Theorem 3.5 (Replication Theorem [44, 52]) *If x occurs in P, Q, and R only in output subject position (as subjects of output prefixes, as negative subjects), then*

$$(\nu x)\,(P \mid Q \mid !x(z).R) \;\approx\; (\nu x)\,(P \mid !x(z).R) \mid (\nu x)\,(Q \mid !x(z).R) \tag{12}$$
$$(\nu x)\,(!P \mid !x(z).R) \;\approx\; !(\nu x)\,(P \mid !x(z).R) \tag{13}$$

and[8] *if x does not occur in π, then*

$$(\nu x)\,(\pi.P \mid !x(z).R) \;\approx\; \pi.(\nu x)\,(P \mid !x(z).R) \tag{14}$$

This permits the '$\approx$' steps in in the proof for Lem. 3.4 when dealing with $M = M_1 M_2$:

$$\begin{array}{rcl}
(\nu x)\,(\llbracket M_1 M_2 \rrbracket^{\mathrm{P}} u \mid !x.\llbracket N \rrbracket^{\mathrm{P}}) & & \\
& \triangleq & (\nu x)\,((\lambda u)\,(\nu v)\,(\llbracket M_1 \rrbracket^{\mathrm{P}} v \mid (\nu z)\,(\overline{v}.[zu] \mid !z.\llbracket M_2 \rrbracket^{\mathrm{P}}))u \mid !x.\llbracket N \rrbracket^{\mathrm{P}}) \\
& = & (\nu x)\,((\nu v)\,(\llbracket M_1 \rrbracket^{\mathrm{P}} v \mid (\nu z)\,(\overline{v}.[zu] \mid !z.\llbracket M_2 \rrbracket^{\mathrm{P}})) \mid !x.\llbracket N \rrbracket^{\mathrm{P}}) \\
(12) & \approx & (\nu v)\,((\nu x)\,(\llbracket M_1 \rrbracket^{\mathrm{P}} v \mid !x.\llbracket N \rrbracket^{\mathrm{P}}) \mid (\nu x)\,((\nu z)\,(\overline{v}.[zu] \mid !z.\llbracket M_2 \rrbracket^{\mathrm{P}}) \mid !x.\llbracket N \rrbracket^{\mathrm{P}})) \\
& \equiv & (\nu v)\,((\nu x)\,(\llbracket M_1 \rrbracket^{\mathrm{P}} v \mid !x.\llbracket N \rrbracket^{\mathrm{P}}) \mid (\nu z)\,(\overline{v}.[zu] \mid (\nu x)\,(!z.\llbracket M_2 \rrbracket^{\mathrm{P}} \mid !x.\llbracket N \rrbracket^{\mathrm{P}}))) \\
(13) & \approx & (\nu v)\,((\nu x)\,(\llbracket M_1 \rrbracket^{\mathrm{P}} v \mid !x.\llbracket N \rrbracket^{\mathrm{P}}) \mid (\nu z)\,(\overline{v}.[zu] \mid !(\nu x)\,(z.\llbracket M_2 \rrbracket^{\mathrm{P}} \mid !x.\llbracket N \rrbracket^{\mathrm{P}}))) \\
(14) & \approx & (\nu v)\,((\nu x)\,(\llbracket M_1 \rrbracket^{\mathrm{P}} v \mid !x.\llbracket N \rrbracket^{\mathrm{P}}) \mid (\nu z)\,(\overline{v}.[zu] \mid !z.(\nu x)\,(\llbracket M_2 \rrbracket^{\mathrm{P}} \mid !x.\llbracket N \rrbracket^{\mathrm{P}}))) \\
(IH) & \approx & (\lambda u)\,(\nu v)\,(\llbracket M_1\{N/x\} \rrbracket^{\mathrm{P}} v \mid (\nu z)\,(\overline{v}.[zu] \mid !z.\llbracket M_2\{N/x\} \rrbracket^{\mathrm{P}}))\,u \\
& \triangleq & \llbracket M_1\{N/x\} M_2\{N/x\} \rrbracket^{\mathrm{P}} u
\end{array}$$

Notice that the induction gets applied for a process $(\nu x)\,(\llbracket M_2 \rrbracket^{\mathrm{P}} \mid !x.\llbracket N \rrbracket^{\mathrm{P}})$ that appears under guard (and replication), *i.e.* in $!z.(\nu x)\,(\llbracket M_2 \rrbracket^{\mathrm{P}} \mid !x.\llbracket N \rrbracket^{\mathrm{P}})$, so Lem. 3.4 is stated using full weak bisimilarity, '$\approx$'. Moreover, the notational differences notwithstanding, the encoding $\llbracket \cdot \rrbracket^{\mathrm{P}} \cdot$ is input-based, and a direct rewrite of the original one.

After Milner's encodings, many variants followed; for example, [52] defines an encoding into the higher-order π-calculus that respects lazy reduction. We repeat that definition here, but adjusted to the normal π-calculus, rather than the higher-order one.

The (call by name) encoding $[\![\cdot]\!]^{\mathcal{N}}$ [9] of the lazy λ-calculus is defined through:

$$\begin{array}{rcll}
[\![x]\!]^{\mathcal{N}} & \triangleq & \overline{x}\,a & \\
[\![\lambda x.M]\!]^{\mathcal{N}} & \triangleq & (\nu v)\,(\overline{a}\,v.v(x,p).[\![M]\!]^{\mathcal{N}}) & (v, p \textit{ fresh}) \\
[\![MN]\!]^{\mathcal{N}} & \triangleq & (\nu q)\,([\![M]\!]^{\mathcal{N}} \mid q(v).(\nu x)\,(\overline{v}\langle x,a\rangle.!x(w).[\![N]\!]^{\mathcal{N}})) & (q, v, x, w \textit{ fresh})
\end{array}$$

Notice that although this is an output-based encoding, in the sense that the (private) channel q in the encoding of MN is used as an output for the encoding of M, underneath the encoding is essentially Milner's. As before, the reductions inside an abstraction, those in the right-hand side of an application, as well as those inside the term that gets substituted cannot be simulated, and therefore this encoding models (part of) lazy reduction.

For this encoding, [52] shows a number of results; first it shows:

$$[\![(\lambda x.M)\,N]\!]^{\mathcal{N}} \;\overset{(1)}{\xrightarrow{\tau}_d{}^{2}}\; (\nu x)\,([\![M]\!]^{\mathcal{N}} \mid !x(w).[\![N]\!]^{\mathcal{N}}) \;\overset{(2)}{\approx_g}\; [\![M\{N/x\}]\!]^{\mathcal{N}}$$

[8] This property is not stated in [44], but is needed, as seen below.

[9] For uniformity of notation, we write $[\![\cdot]\!]^{\mathcal{N}}$ rather than $\mathcal{N}[\![\cdot]\!]\cdot$.

(here '$\xrightarrow{\tau}_d$' is the deterministic (silent) transition and '$\approx_g$' is ground bisimilarity) which leads to:[10]

$$M \to_{\mathrm{L}} N \overset{(3)}{\Rightarrow} [\![M]\!]^{\mathcal{N}} \xrightarrow{\tau}_d^2 \approx_g [\![N]\!]^{\mathcal{N}} \qquad M =_\beta N \overset{(4)}{\Rightarrow} [\![M]\!]^{\mathcal{N}} \approx_g [\![N]\!]^{\mathcal{N}}$$

As in [44], a variant of Lemma 2.5 is needed to achieve this result. We show the equivalent of these results for our encoding in Thm 7.1 and Thm 7.7 below.

The characterisation of $\llbracket M \rrbracket^{\mathrm{M}} a \approx \llbracket N \rrbracket^{\mathrm{M}} a$, left as open problem in [43], was achieved through showing that

$$\llbracket M \rrbracket^{\mathrm{M}} a \approx \llbracket N \rrbracket^{\mathrm{M}} a \iff M \simeq N,$$

where '$\simeq$' is the *applicative bisimilarity* on λ-terms, an operational notion of equivalence on terms of the lazy λ-calculus as defined by Abramsky and Ong [4], rather than β-equality.

This result comes with caveats, however: as shown by Ong [46], applicative bisimulation equates $x(x\Theta\Delta\Delta)\Theta$ and $x(\lambda y.x\Theta\Delta\Delta y)\Theta$ (where $\Delta = \lambda x.xx$, and Θ is such that, for every N, ΘN is reducible to an abstraction) whereas these terms are not weakly bisimilar under the interpretation $\llbracket \cdot \rrbracket^{\mathrm{M}} \cdot$ (see [43]). This has strong repercussion as far as the interpretation of the λ-calculus is concerned: in order to achieve full abstraction, Sangiorgi had to extend Milner's encoding to Λ_c, a λ-calculus enriched with constants (that take the place of the free variables, thereby creating closed terms) and by exploiting a more abstract encoding into the *Higher Order* π-calculus, a variant of the π-calculus with higher-order communications.

Sangiorgi's result then essentially states that the interpretations of closed Λ_c-terms M and N are weakly bisimilar if and only if M and N are applicatively bisimilar; in [51] he improves on this by showing that the interpretation of terms M and N in Λ_c in the standard π-calculus is weakly bisimilar if and only if M and N have the same Lévy-Longo tree [40, 42] (a lazy variant of Böhm trees [15]). Since the principal results in [51], presented almost all without proof, are shown for closed terms only, [11] Sangiorgi's full abstraction result only deals with closed terms.

An output-based encoding for the λ-calculus

In [12] we presented a *logical*, *output-based* spine interpretation $\llbracket \cdot \rrbracket^{\mathrm{s}} \cdot$ that interprets abstraction $\lambda x.M$ not using *input*, but via an asynchronous *output* which leaves the interpretation of the body M free to reduce. That interpretation is defined as:

$$\begin{array}{rcll}
\llbracket x \rrbracket^{\mathrm{s}} a & \triangleq & x(w).\overline{a}\,w & (w \textit{ fresh}) \\
\llbracket \lambda x.M \rrbracket^{\mathrm{s}} a & \triangleq & (\nu xb)\,(\llbracket M \rrbracket^{\mathrm{s}} b \mid \overline{a}\langle x,b\rangle) & (b \textit{ fresh}) \\
\llbracket MN \rrbracket^{\mathrm{s}} a & \triangleq & (\nu c)\,(\llbracket M \rrbracket^{\mathrm{s}} c \mid c(v,d).(\llbracket v := N \rrbracket^{\mathrm{s}} \mid d \rightarrow \overline{a})) & (c,v,d \textit{ fresh}) \\
\llbracket M\langle x := N\rangle \rrbracket^{\mathrm{s}} a & \triangleq & (\nu x)\,(\llbracket M \rrbracket^{\mathrm{s}} a \mid \llbracket x := N \rrbracket^{\mathrm{s}}) & \\
\llbracket x := N \rrbracket^{\mathrm{s}} & \triangleq & !\llbracket N \rrbracket^{\mathrm{s}} x &
\end{array}$$

This interpretation is directly extracted from the embedding of the λ-calculus into $\mathcal{X}$, as discussed in [11], and the embedding of $\mathcal{X}$ into the π-calculus, as defined in [10]. Since the first of these embeddings maps proofs to proofs, the encoding is called logical.

This can be seen as a variant of one defined by Beffara [18], obtained through linear logic, except for the input/output polarity for variables.

For this interpretation, [12] showed Operational Soundness and Type Preservation, but with respect to the notion of *explicit head-reduction* '$\to_{\mathrm{XH}}$', similar to the notion defined below in Def. 5.1, and the notion of type assignment '$\vdash_\pi$' defined in [12]. The main results shown are:

[10] In [52], it is suggested that (4) follows from (3), but in fact it follows from (1) and (2). Moreover, we assume that the formulation of (4), where '$\cong^c$' is used instead of '$\approx_g$', is a typo.

[11] The development of Lévy-Longo trees is done for all terms, but the build-up of the main result Thm 5.4 includes Thm 4.11 that holds, other than suggested, only for closed terms.

i) If $M\Uparrow$ then $\llbracket M \rrbracket^{s}_{\pi} a \Uparrow_{\pi}$, and if $M \to_{\mathrm{XH}} N$ then $\llbracket M \rrbracket^{s}_{\pi} a \to^{*}_{\pi} \sim_{\mathrm{C}} \llbracket N \rrbracket^{\mathrm{H}}_{\pi} a$.

ii) If $\Gamma \vdash M : A$ then $\llbracket M \rrbracket^{s}_{\pi} a : \Gamma \vdash_{\pi} a{:}A$.

where '$\sim_{\mathrm{C}}$' is contextual equivalence,

As argued in [12], to show this result, which formulates a direct *step-by-step* relation between β-reduction and the synchronisation in the π-calculus, it was necessary to make the substitution explicit. This is a direct consequence of the fact that, in the π-calculus, the implicit substitution of the λ-calculus gets 'implemented' *one variable at the time*, rather than all in one fell swoop. Since we aim to show a similar result for $\lambda\mu$, we will therefore also here define a notion of explicit substitution for that calculus (see the next section).

Classical logic and the π-calculus

There are, to date, a number of papers that investigate if the π-calculus can be used to interpret calculi that relate to classical logic as well, like $\lambda\mu$, $\lambda\mu\tilde{\mu}$, or $\mathcal{X}$.

In [36] an interpretation of Call-by-Value $\lambda\mu$ is defined that is based on Milner's, but allows for a much more liberal notion of reduction on processes, and considers fully-typed terms (so types are part of the syntax of terms) and processes only. Types for processes prescribe usage of names, and name passing is restricted to *bound (private, hidden) name passing*.[12] The syntax of processes considered there is

$$P \ ::= \ !x(\vec{y}).P \ \mid \ (\nu\vec{y})\,(\overline{x}\,\vec{y} \mid P) \ \mid \ P \mid Q \ \mid \ (\nu x)\,P \ \mid \ 0$$

and the notion of reduction on processes is extended to that of '$\searrow$', defined as the least compatible relation over typed processes (*i.e.* closed under typed contexts), taken modulo '$\equiv$', that includes:

$$!x(\vec{y}).P \mid (\nu\vec{a})\,(\overline{x}\,\vec{a} \mid Q) \ \to \ !x(\vec{y}).P \mid (\nu\vec{a})\,(P\{\overrightarrow{a/y}\} \mid Q)$$

as the basic synchronisation rule, as well as

$$\begin{array}{rcl} \mathsf{C}\lceil(\nu\vec{a})\,(\overline{x}\,\vec{a} \mid P)\rfloor \mid !x(\vec{y}).Q & \searrow_{r} & \mathsf{C}\lceil(\nu\vec{a})\,(P\{\overrightarrow{a/y}\} \mid Q)\rfloor \mid !x(\vec{y}).Q \\ (\nu x)\,(!x(\vec{y}).Q) & \searrow_{g} & 0 \end{array}$$

where $\mathsf{C}\lceil\cdot\rfloor$ is an arbitrary (typed) context; note that '$\searrow$' synchronises with any occurrence of $\overline{x}\,\vec{a}$, no matter what guards they may be placed under. The resulting calculus is thereby very different from the original π-calculus.

On the relation between Girard's linear logic [31] and the π-calculus, [19] gives a treatment of information flow in proof-nets; only a small fragment of Linear Logic was considered, and the translation between proofs and π-calculus was left rather implicit, as also noted in [23]. To illustrate this, [19] uses the standard syntax for the polyadic π-calculus

$$P, Q \ ::= \ 0 \mid P \mid Q \mid !P \mid (\nu a)\,P \mid a(\vec{x}).P \mid \overline{a}\,\vec{p}.P$$

similar to the one we use here (see Def. 2.1) but for the fact that in [19] the *let*-construct is not used. However, the encoding of a 'cut' in linear logic

$$\dfrac{\dfrac{}{\vdash x{:}A \otimes B, y{:}(A \otimes B)^{\perp}} \qquad \dfrac{\dfrac{}{\vdash n{:}A, m{:}A^{\perp}} \qquad \dfrac{}{\vdash z{:}B, w{:}B^{\perp}}}{\vdash m{:}A^{\perp}, w{:}B^{\perp}, v{:}A \otimes B}}{\vdash x{:}A \otimes B, m{:}A^{\perp}, w{:}B^{\perp}}$$

i.e. the 'term' $x{:}A \otimes B, m{:}A^{\perp}, w{:}B^{\perp}$, gets translated into a 'language of proofs', the result of which looks like:

$$Cut^{k}(I, \textstyle\bigotimes_{v}^{n,z}(I,I)mwz)x,(m,w) \ = \ (\nu k)(I\{k/y\} \mid \textstyle\bigotimes_{v}^{n,z}(I,I)mwz\{k/v\})$$

[12] This is a feature of all related interpretations into the π-calculus.

where the terms Cut and I are (rather loosely) defined. Notice the use of arbitrary application of processes to channel names, and the operation of pairing; the authors do not specify how to relate this notation, and in particular their notion of application of process names without adding Milner's abstraction mechanism explicitly, to the above (application free) syntax of processes they consider.

However, even if this relationship is made explicit, also then a different π-calculus is needed to make the encoding work. To clarify this point, consider the translation in the π-calculus of the term above, which according to the definition given in [19] becomes:

$$(\nu k)\,(x(a).k(a) \mid (\nu nz)(\overline{k}(n,z).(n(b).m(b) \mid z(b).w(b)))).$$

Although intended, no communication is possible in this term, as the arity of the channel k does not match. To overcome this kind of problem, Bellin and Scott would need the *let*-construct with use of pairs of names as we have introduced in this paper in Def. 2.1.

In [10] an interpretation into π of the sequent calculus $\mathcal{X}$ is defined that enjoys the Curry-Howard isomorphism for Gentzen's LK, which is shown to respect reduction. However, this result is only partial, as it is formulated as '*if* $P \to_{\mathcal{X}} Q$, *then* $[\![P]\!] \,{}_{\mathsf{c}}\!\sqsupseteq [\![Q]\!]$', allowing $[\![P]\!]$ to have more observable behaviour than $[\![Q]\!]$. The main reason for this is that reduction in $\mathcal{X}$ is non-confluent and that the π-calculus has no feature to erase part of a process that is not garbage. Taking the representation of the *cut*, $P\alpha \dagger xQ$, assume α not in P and x not in Q, then $P\alpha \dagger xQ \to_{\mathcal{X}} P$ and $P\alpha \dagger xQ \to_{\mathcal{X}} Q$, and $[\![P\alpha \dagger xQ]\!] = [\![P]\!] \mid [\![Q]\!]$. Both P and Q are valid results under reduction in $\mathcal{X}$, so should be represented in the encoding without preference for either; moreover, both could be normal forms (both witnesses of the same sequent), that can each interact with a context, so it is not guaranteed that either $[\![P]\!]$ or $[\![Q]\!]$ can be considered garbage. In [10] it is argued that this is natural in the context of non-confluent, symmetric sequent calculi.

An interpretation of $\lambda\mu\tilde{\mu}$ is studied in [25]; the interpretation defined there strongly depends on recursion, is not compositional, and preserves only outermost reduction; no relation with types is shown.

4 $\lambda\mu\mathbf{x}$: $\lambda\mu$ with explicit substitution

One of the main achievements of [12] is that it establishes a strong link between reduction in the π-calculus and step-by-step *explicit substitution* for the λ-calculus, as in $\lambda\mathbf{x}$ [20], by formulating a result not only with respect to explicit head-reduction and the spine interpretation, but also for Milner's interpretation [43] with respect to explicit lazy reduction (see also Thm A.7), all defined in [12]. In view of this, for the purpose of defining an interpretation for $\Lambda\mu$ into the π-calculus in [13], it was natural to study a variant of $\Lambda\mu$ with explicit substitution as well; since here we work with $\lambda\mu$, here we present $\lambda\mu\mathbf{x}$.

Explicit substitution treats substitution as a first-class operator, both for the logical and the structural substitution, and describes all the necessary steps to effectuate both. A difference between Krivine's machine and $\lambda\mathbf{x}$ is that substitutions $\langle x{:=}N\rangle$ are part of term construction, not stored separately in an environment. They are created by redex contraction, through the step $\mathsf{C}\lceil(\lambda x.M)\,N\rfloor \to \mathsf{C}\lceil M\langle x{:=}N\rangle\rfloor$, so the substitution gets applied directly to the term, and the newly created term gets placed at the location of the redex. Moreover, in $\lambda\mathbf{x}$ reduction is free, and substitutions can appear throughout the term.

Definition 4.1 ($\lambda\mu\mathbf{x}$) *i*) Terms of $\lambda\mu\mathbf{x}$, the *$\lambda\mu$ calculus with explicit substitution*, are defined through the grammar:

$$M,N \;::=\; x \mid \lambda x.M \mid MN \mid M\langle x{:=}N\rangle \mid \mu\alpha.[\beta]M \mid M\langle \alpha{:=}N{\cdot}\gamma\rangle$$

where x ranges over a infinite countable set of *variables*, and a, β and γ range over a infinite countable set of *names*.

ii) *Bound variables and names* of terms are defined by:

$$\begin{array}{lcllcl}
bv(x) & = & \emptyset & bn(x) & = & \emptyset \\
bv(\lambda x.M) & = & bv(M) \cup \{x\} & bn(\lambda x.M) & = & bn(M) \\
bv(MN) & = & bv(M) \cup bv(N) & bn(MN) & = & bn(M) \cup bn(N) \\
bv(M\langle x{:=}N\rangle) & = & bv(M) \cup \{x\} \cup bv(N) & bn(M\langle x{:=}N\rangle) & = & bn(M) \cup bn(N) \\
bv(\mu\alpha.[\beta]M) & = & bv(M) & bn(\mu\alpha.[\beta]M) & = & bn(M) \cup \{\alpha\} \\
bv(M\langle \alpha{:=}N{\cdot}\gamma\rangle) & = & bv(M) \cup bv(N) & bn(M\langle \alpha{:=}N{\cdot}\gamma\rangle) & = & bn(M) \cup \{\alpha\} \cup bn(N)
\end{array}$$

and we call a variable x or name α *free* in M (and write $x \in fv(M)$ and $\alpha \in fn(M)$) if it occurs in M and is not bound. We use Barendregt's convention, that demands that free and bound names and variables are distinct; then when using $M\langle x{:=}N\rangle$ and $M\langle \alpha{:=}N{\cdot}\gamma\rangle$, we can assume that x and α do not appear outside M. [13]

iii) We call a term $M \in \lambda\mu\mathbf{x}$ *pure* if M contains no explicit substitutions, so if $M \in \lambda\mu$.

iv) The reduction relation '$\to_{\mathrm{x}}$' on terms in $\lambda\mu\mathbf{x}$ is defined through the following rules (for the sake of completeness, we list all):

Main reduction rules:

$$\begin{array}{rcll}
(\lambda x.M)N & \to & M\langle x{:=}N\rangle & \\
(\mu\alpha.[\alpha]M)N & \to & \mu\gamma.[\gamma](M\langle \alpha{:=}N{\cdot}\gamma\rangle)N & (\gamma \textit{ fresh}) \\
(\mu\alpha.[\beta]M)N & \to & \mu\gamma.[\beta](M\langle \alpha{:=}N{\cdot}\gamma\rangle) & (\alpha \neq \beta,\ \gamma \textit{ fresh}) \\
\mu\beta.[\beta]M & \to & M & (\beta \notin fn(M)) \\
\mu\alpha.[\beta]\mu\gamma.[\delta]M & \to & \begin{cases} \mu\alpha.[\beta]M\{\beta/\gamma\} \ ^{14} & (\delta = \gamma) \\ \mu\alpha.[\delta]M\{\beta/\gamma\} & (\delta \neq \gamma) \end{cases} &
\end{array}$$

Term substitution rules:

$$\begin{array}{rcll}
x\langle x{:=}N\rangle & \to & N & \\
y\langle x{:=}N\rangle & \to & M & (x \neq y) \\
(\lambda y.M)\langle x{:=}N\rangle & \to & \lambda y.(M\langle x{:=}N\rangle) & \\
(PQ)\langle x{:=}N\rangle & \to & (P\langle x{:=}N\rangle)(Q\langle x{:=}N\rangle) & \\
(\mu\alpha.[\beta]M)\langle x{:=}N\rangle & \to & \mu\alpha.[\beta](M\langle x{:=}N\rangle) &
\end{array}$$

Structural substitution rules:

$$\begin{array}{rcll}
M\langle \alpha{:=}N{\cdot}\gamma\rangle & \to & M & (\alpha \notin fn(M)) \\
(\mu\delta.[\alpha]M)\langle \alpha{:=}N{\cdot}\gamma\rangle & \to & \mu\delta.[\gamma]M\langle \alpha{:=}N{\cdot}\gamma\rangle N & \\
(\mu\delta.[\beta]M)\langle \alpha{:=}N{\cdot}\gamma\rangle & \to & \mu\delta.[\beta]M\langle \alpha{:=}N{\cdot}\gamma\rangle & \\
(\lambda x.M)\langle \alpha{:=}N{\cdot}\gamma\rangle & \to & \lambda x.M\langle \alpha{:=}N{\cdot}\gamma\rangle & \\
(PQ)\langle \alpha{:=}N{\cdot}\gamma\rangle & \to & (P\langle \alpha{:=}N{\cdot}\gamma\rangle)(Q\langle \alpha{:=}N{\cdot}\gamma\rangle) &
\end{array}$$

Contextual rules:

$$M \to N \ \Rightarrow \ \begin{cases} \lambda x.M & \to \ \lambda x.N \\ ML & \to \ NL \\ LM & \to \ LN \\ \mu\alpha.[\beta]M & \to \ \mu\alpha.[\beta]N \end{cases} \qquad M \to N \ \Rightarrow \ \begin{cases} M\langle x{:=}L\rangle & \to \ N\langle x{:=}L\rangle \\ L\langle x{:=}M\rangle & \to \ L\langle x{:=}N\rangle \\ M\langle \alpha{:=}L{\cdot}\gamma\rangle & \to \ N\langle \alpha{:=}L{\cdot}\gamma\rangle \\ L\langle \alpha{:=}M{\cdot}\gamma\rangle & \to \ L\langle \alpha{:=}N{\cdot}\gamma\rangle \end{cases}$$

v) We use '$\to_{:=}$' for the notion of reduction where only term substitution, structural, or contextual rules are used (so not the main reduction rules), and '$=_{\mathrm{x}}$' for the congruence generated by '$\to_{\mathrm{x}}$'.

Notice that we do not add rules like

$$\begin{array}{rcl}
M\langle x{:=}N\rangle\langle y{:=}L\rangle & \to & M\langle y{:=}L\rangle\langle x{:=}N\langle y{:=}L\rangle\rangle \\
M\langle x{:=}N\rangle\langle y{:=}L\rangle & \to & M\langle y{:=}L\rangle\langle x{:=}N\rangle\langle y{:=}L\rangle
\end{array}$$

[13] Note that here, for the explicit case, the convention to 'reuse' α rather than introduce the new name γ would create a violation of Barendregt's convention.

[14] Notice that this alternative is defined using renaming; since β itself is not a term, we cannot use explicit substitution for this operation.

since as in [20], this would introduce undesired non-termination. Moreover, since reduction in $\lambda\mu\mathbf{x}$ actually is formulated via term rewriting rules [37], reduction is allowed to take place also *inside the substitution term*, before the actual substitution takes place.

This notion of $\lambda\mu$ with explicit substitution differs from that of [6], where a version with explicit substitution is defined for a variant of $\lambda\mu$ that uses de Bruijn indices [22].

Notice that, as a result of reduction, substitutions can appear inside applications, as occurs in:

$$\begin{array}{rcl}(\lambda x_3.(\lambda x_2.(\lambda x_1.y)N_1M_2)N_2M_3)N_3 & \to_{\mathrm{xH}} & ((\lambda x_2.(\lambda x_1.y)N_1M_2)N_2M_3)\langle x_3{:=}N_3\rangle \\ & \to_{\mathrm{xH}} & (((\lambda x_1.y)N_1M_2)\langle x_2{:=}N_2\rangle M_3)\langle x_3{:=}N_3\rangle \\ & \to_{\mathrm{xH}} & ((y\langle x_1{:=}N_1\rangle M_2)\langle x_2{:=}N_2\rangle M_3)\langle x_3{:=}N_3\rangle\end{array}$$

(we write the latter term as $y\langle x_1{:=}N_1\rangle M_2\langle x_2{:=}N_2\rangle M_3\langle x_3{:=}N_3\rangle$).

The following is easy to show:

Proposition 4.2 The rule $M\langle x{:=}N\rangle \to M$ if $x \notin fv(M)$ is admissible.

Explicit substitution describes explicitly the process of executing a $\beta\mu$-reduction, *i.e.* expresses syntactically the details of the computation as a succession of atomic steps (like in a first-order rewriting system), where the implicit substitution of each $\beta\mu$-reduction step is split up into reduction steps. Thereby the following is straightforward:

Proposition 4.3 ($\lambda\mu\mathbf{x}$ IMPLEMENTS $\lambda\mu$-REDUCTION) *$M \to_{\beta\mu} N \Rightarrow M \to_{\mathrm{x}}^{*} N$.*

Proof: Straightforward. □

Type assignment on $\lambda\mu\mathbf{x}$ is a natural extension of the system of Def. 1.2 by adding rules (*T-sub*) and (*C-sub*).

Definition 4.4 (TYPE ASSIGNMENT FOR $\lambda\mu\mathbf{x}$) Using the notion of types in Def. 1.2, type assignment for $\lambda\mu\mathbf{x}$ is defined by:

$$(Ax):\ \frac{}{\Gamma,x{:}A \vdash x : A \mid \Delta} \qquad (\mu):\ \frac{\Gamma \vdash M : B \mid \alpha{:}A,\Delta}{\Gamma \vdash \mu\alpha.[\beta]M : A \mid \beta{:}B,\Delta}\ (\alpha \notin \Delta) \quad \frac{\Gamma \vdash M : A \mid \alpha{:}A,\Delta}{\Gamma \vdash \mu\alpha.[\alpha]M : A \mid \Delta}\ (\alpha \notin \Delta)$$

$$(\to\!I):\ \frac{\Gamma,x{:}A \vdash M : B \mid \Delta}{\Gamma \vdash \lambda x.M : A \to B \mid \Delta}\ (x \notin \Gamma) \qquad (T\text{-}sub):\ \frac{\Gamma,x{:}A \vdash M : B \mid \Delta \quad \Gamma \vdash N : A \mid \Delta}{\Gamma \vdash M\langle x{:=}N\rangle : B \mid \Delta}\ (x \notin \Gamma)$$

$$(\to\!E):\ \frac{\Gamma \vdash M : A \to B \mid \Delta \quad \Gamma \vdash N : A \mid \Delta}{\Gamma \vdash MN : B \mid \Delta} \qquad (C\text{-}sub):\ \frac{\Gamma \vdash M : C \mid \alpha{:}A \to B,\Delta \quad \Gamma \vdash N : A \mid \Delta}{\Gamma \vdash M\langle \alpha{:=}N{\cdot}\gamma\rangle : C \mid \gamma{:}B,\Delta}\ (\alpha,\gamma \notin \Delta)$$

We write $\Gamma \vdash_{\mu\mathrm{x}} M : A \mid \Delta$ for judgements derivable in this system.

For this notion of type assignment, having extended $\lambda\mu$, we need to show the usual soundness result (*i.e* assignable types are preserved under reduction), for which we first need to show the admissibility of thinning and weakening.

Lemma 4.5 (WEAKENING AND THINNING) *Weakening: If $\Gamma \vdash_{\mu\mathrm{x}} M : A \mid \Delta$ and $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$, then $\Gamma' \vdash_{\mu\mathrm{x}} M : A \mid \Delta'$.*

Thinning: If $\Gamma \vdash_{\mu\mathrm{x}} M : A \mid \Delta$ and $\Gamma' = \{\, x{:}B \in \Gamma \mid x \in fv(M) \,\}$ and $\Delta' = \{\, \alpha{:}B \in \Delta \mid \alpha \in fn(M) \,\}$, then $\Gamma' \vdash_{\mu\mathrm{x}} M : A \mid \Delta'$.

Proof: Straightforward. □

We can now show:

Theorem 4.6 (SUBJECT REDUCTION) *If $P \to_{\mathrm{x}} Q$, and $\Gamma \vdash_{\mu\mathrm{x}} P : A \mid \Delta$ then $\Gamma \vdash_{\mu\mathrm{x}} Q : A \mid \Delta$.*

Proof: We show the result for a selection of the reduction rules.

$(\lambda x.M)\,N \to M\langle x{:=}N\rangle$: Then the derivation is shaped like the derivation on the left, from which we can construct the one on the right.

$$\dfrac{\dfrac{\boxed{\phantom{\Gamma,x{:}B \vdash M : A \mid \Delta}}}{\Gamma \vdash \lambda x.M : B{\to}A \mid \Delta}\,(\to I) \qquad \boxed{}}{\Gamma \vdash (\lambda x.M)\,N : A \mid \Delta}\,(\to E) \qquad\qquad \dfrac{\boxed{\phantom{\Gamma,x{:}B \vdash M : A \mid \Delta}} \quad \boxed{}}{\Gamma \vdash M\langle x{:=}N\rangle : A \mid \Delta}\,(\textit{T-sub})$$

$(\mu\alpha.\mathrm{C})N \to \mu\gamma.\mathrm{C}\langle\alpha{:=}N{\cdot}\gamma\rangle$: We have two cases:

$\mathrm{C} = [\alpha]M$: Then $\mu\gamma.([\alpha]M)\,\langle\alpha{:=}N{\cdot}\gamma\rangle = \mu\gamma.[\gamma]\,(M\langle\alpha{:=}N{\cdot}\gamma\rangle)N$, and the derivation is shaped like

$$\dfrac{\dfrac{\boxed{\phantom{\Gamma \vdash M : B{\to}A \mid \alpha{:}B{\to}A,\Delta}}}{\Gamma \vdash \mu\alpha.[\alpha]M : B{\to}A \mid \Delta}\,(\mu) \qquad \boxed{}}{\Gamma \vdash (\mu\alpha.[\alpha]M)N : A \mid \Delta}\,(\to E)$$

from which we can construct

$$\dfrac{\dfrac{\dfrac{\boxed{\phantom{\Gamma \vdash M : B{\to}A \mid \alpha{:}B{\to}A,\Delta}} \quad \boxed{}}{\Gamma \vdash M\langle\alpha{:=}N{\cdot}\gamma\rangle : B{\to}A \mid \gamma{:}A,\Delta}\,(\textit{C-sub}) \qquad \dfrac{\boxed{}}{\Gamma \vdash N : B \mid \gamma{:}A,\Delta}\,(\textit{Weak})}{\Gamma \vdash (M\langle\alpha{:=}N{\cdot}\gamma\rangle)N : A \mid \gamma{:}A,\Delta}\,(\to E)}{\Gamma \vdash \mu\gamma.[\gamma](M\langle\alpha{:=}N{\cdot}\gamma\rangle)N : A \mid \Delta}\,(\mu)$$

$\mathrm{C} = [\beta]M$, *with* $\alpha \neq \beta$: Then $\mu\gamma.([\beta]M)\,\langle\alpha{:=}N{\cdot}\gamma\rangle = \mu\gamma.[\beta]M\langle\alpha{:=}N{\cdot}\gamma\rangle$. Then the derivation is shaped like

$$\dfrac{\dfrac{\boxed{\phantom{\Gamma \vdash M : C \mid \alpha{:}A{\to}B,\beta{:}C,\Delta}}}{\Gamma \vdash \mu\alpha.[\beta]M : A{\to}B \mid \beta{:}C,\Delta}\,(\mu) \qquad \boxed{\phantom{\Gamma \vdash N : A \mid \beta{:}C,\Delta}}}{\Gamma \vdash (\mu\alpha.[\beta]M)N : B \mid \beta{:}C,\Delta}\,(\to E)$$

from which we can construct

$$\dfrac{\dfrac{\boxed{\phantom{\Gamma \vdash_{\mu\mathrm{x}} M : C \mid \alpha{:}A{\to}B,\beta{:}C,\Delta}} \quad \boxed{\phantom{\Gamma \vdash_{\mu\mathrm{x}} N : A \mid \beta{:}C,\Delta}}}{\Gamma \vdash_{\mu\mathrm{x}} M\langle\alpha{:=}N{\cdot}\gamma\rangle : C \mid \gamma{:}B,\beta{:}C,\Delta}\,(\textit{C-sub})}{\Gamma \vdash_{\mu\mathrm{x}} \mu\gamma.[\beta](M\langle\alpha{:=}N{\cdot}\gamma\rangle) : B \mid \beta{:}C,\Delta}\,(\mu)$$

since β and γ have the same type.

$\mu\delta.[\beta](\mu\gamma.[\alpha]M) \to \mu\delta.[\alpha]M\{\beta/\gamma\}$: (We assume all names are distinct; if not, the proof is similar.) Then the derivation is shaped like the derivation on the left, from which we can construct the one on the right.

$$\dfrac{\dfrac{\boxed{\phantom{\Gamma \vdash M : C \mid \delta{:}A,\gamma{:}B,\beta{:}B,\alpha{:}C,\Delta}}}{\Gamma \vdash \mu\gamma.[\alpha]M : B \mid \delta{:}A,\beta{:}B,\alpha{:}C,\Delta}\,(\mu)}{\Gamma \vdash \mu\delta.[\beta](\mu\gamma.[\alpha]M) : A \mid \beta{:}B,\alpha{:}C,\Delta}\,(\mu) \qquad\qquad \dfrac{\boxed{\phantom{\Gamma \vdash M\{\beta/\gamma\} : C \mid \delta{:}A,\beta{:}B,\alpha{:}C,\Delta}}}{\Gamma \vdash \mu\delta.[\alpha]M\{\beta/\gamma\} : A \mid \beta{:}B,\alpha{:}C,\Delta}\,(\mu)$$

$\mu\alpha.[\alpha]M \to M$, *if* $\alpha \notin fn(M)$: Then the derivation is shaped like

$$\dfrac{\boxed{\phantom{\Gamma \vdash M : A \mid \alpha{:}C,\Delta}}}{\Gamma \vdash \mu\alpha.[\alpha]M : A \mid \Delta}\,(\mu)$$

Since $\alpha \notin fn(M)$, by thinning also $\Gamma \vdash_{\mu\mathrm{x}} M : A \mid \Delta$.

$M\langle x{:=}N\rangle \to M$, *if* $x \notin fv(M)$: Then the derivation is shaped like

$$\dfrac{\boxed{\phantom{\Gamma, x{:}B \vdash M : A \mid \Delta}} \qquad \boxed{}}{\Gamma \vdash M\langle x{:=}N\rangle : A \mid \Delta}\ (\textit{T-sub})$$

Since $x \notin fv(M)$, by thinning from the left-hand sub-derivation also $\Gamma \vdash_{\mu\mathrm{x}} M : A \mid \Delta$.

$(\mu\beta.[\alpha]M)\langle \alpha{:=}N{\cdot}\gamma\rangle \to \mu\beta.[\gamma](M\langle \alpha{:=}N{\cdot}\gamma\rangle)N$: Then the derivation is shaped like

$$\dfrac{\dfrac{\boxed{\phantom{\Gamma \vdash M : B{\to}C \mid \alpha{:}B{\to}C, \beta{:}A, \Delta}}}{\Gamma \vdash \mu\beta.[\alpha]M : A \mid \alpha{:}B{\to}C, \Delta}\ (\mu) \qquad \boxed{}}{\Gamma \vdash (\mu\beta.[\alpha]M)\langle \alpha{:=}N{\cdot}\gamma\rangle : A \mid \gamma{:}C, \Delta}\ (\textit{C-sub})$$

from which we can construct:

$$\dfrac{\dfrac{\dfrac{\boxed{\phantom{\Gamma \vdash M : B{\to}C \mid \alpha{:}B{\to}C, \beta{:}A, \Delta}}} \qquad \dfrac{\boxed{}}{\Gamma \vdash N : B \mid \beta{:}A, \Delta}\ (\textit{Weak})}{\Gamma \vdash M\langle \alpha{:=}N{\cdot}\gamma\rangle : B{\to}C \mid \gamma{:}C, \beta{:}A, \Delta}\ (\textit{C-sub}) \qquad \dfrac{\boxed{}}{\Gamma \vdash N : B \mid \gamma{:}C, \beta{:}A, \Delta}\ (\textit{Weak})}{\Gamma \vdash (M\langle \alpha{:=}N{\cdot}\gamma\rangle)N : C \mid \gamma{:}C, \beta{:}A, \Delta}\ (\to E)$$
$$\dfrac{\Gamma \vdash (M\langle \alpha{:=}N{\cdot}\gamma\rangle)N : C \mid \gamma{:}C, \beta{:}A, \Delta}{\Gamma \vdash \mu\beta.[\gamma](M\langle \alpha{:=}N{\cdot}\gamma\rangle)N : A \mid \gamma{:}C, \Delta}\ (\mu)$$

□

5 Explicit head-reduction

In the context of head reduction and explicit substitution, we can economise further on how substitution is executed, and perform only those that are essential for the continuation of head reduction. We will therefore limit substitution to allow it to *only replace* the head variable of a term (this principle is also found in Krivine's machine) or perform a contextual substitution only on names that occur in front of the term. We borrow from Krivine's machine the idea that substitutions are treated as *closures*, a resource; while in '$\to_{\mathrm{x}}$', a substitution $\langle x{:=}N\rangle$ has to be brought to individual variables before coming applicable, for explicit head-reduction '$\to_{\mathrm{xH}}$' as we will define in this section, we essentially choose to have it change $xM_1 \cdots M_n\langle x{:=}N\rangle$ into $NM_1 \cdots M_n\langle x{:=}N\rangle$, replacing only the head variable, and keeping the substitution in its place, where it was created by redex contraction. The results of [12] show that this is exactly the kind of reduction that the π-calculus naturally encodes, which we will confirm again here.

Definition 5.1 (Explicit head-reduction) We define *explicit head-reduction* '$\to_{\mathrm{xH}}$' on $\lambda\mu\mathbf{x}$ as '$\to_{\mathrm{x}}$', but change, remove, and add a few rules:

i) The set of terms is defined as in Def. 4.1. To ease notation, we will use $\boldsymbol{S}$ for a sequence (possibly empty) of substitutions of the shape $\langle x{:=}N\rangle$ or $\langle \alpha{:=}N{\cdot}\gamma\rangle$ when the exact contents of the substitutions is not relevant. We write $x \in \boldsymbol{S}$ if $\langle x{:=}N\rangle \in \boldsymbol{S}$ for some N, and say that $\boldsymbol{S}$ is *defined on* x, and write $\boldsymbol{S}\backslash x$ for $\boldsymbol{S}_1\boldsymbol{S}_2$ if $\boldsymbol{S} = \boldsymbol{S}_1\langle x{:=}N\rangle\boldsymbol{S}_2$ and similarly for $\alpha \in \boldsymbol{S}$ and $\boldsymbol{S}\backslash\alpha$. The variables and names that are defined in $\boldsymbol{S}$ are unique. We write $\boldsymbol{S}_c$ if $\boldsymbol{S}$ is *only* defined on c, *i.e.* $\boldsymbol{S}_c = \langle c{:=}N\rangle$ or $\boldsymbol{S}_c = \langle c{:=}N{\cdot}\gamma\rangle$, and $M_1\boldsymbol{S}_1M_2\boldsymbol{S}_2$ for $((M_1\boldsymbol{S}_1)M_2)\boldsymbol{S}_2$.

ii) The main reduction rules are as before:

$$\begin{array}{rrcll}
(\beta): & (\lambda x.M)N & \to & M\langle x{:=}N\rangle & (\beta\textit{-rule}) \\
(\mu_p): & (\mu\alpha.[\alpha]M)N & \to & \mu\gamma.[\gamma](M\langle \alpha{:=}N{\cdot}\gamma\rangle)N & (\gamma\ \textit{fresh}) \\
(\mu_r): & (\mu\alpha.[\beta]M)N & \to & \mu\gamma.[\beta](M\langle \alpha{:=}N{\cdot}\gamma\rangle) & (\alpha \neq \beta,\ \gamma\ \textit{fresh}) \\
(R): & \mu\alpha.[\beta]\mu\gamma.[\delta]M & \to & \begin{cases} \mu\alpha.[\beta]M\{\beta/\gamma\} & (\delta = \gamma) \\ \mu\alpha.[\delta]M\{\beta/\gamma\} & (\delta \neq \gamma) \end{cases} & \\
(C): & \mu\alpha.[\alpha]M & \to & M & (\alpha \notin fn(M))
\end{array}$$

iii) We combine the substitution rules, and replace the rule for application and term variables:

$$\begin{array}{rrcll}
(hv): & x\boldsymbol{S}_0 M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n \langle x{:=}N\rangle & \to & N\boldsymbol{S}_0 M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n \langle x{:=}N\rangle^{15} & (n \geq 0) \\
(\lambda\boldsymbol{S}): & (\lambda y.M)\,\boldsymbol{S} & \to & \lambda y.(M\boldsymbol{S}) & \\
(hn): & (\mu\delta.[\alpha]M)\,\boldsymbol{S}\langle\alpha{:=}N{\cdot}\gamma\rangle & \to & \mu\delta.[\gamma](M\boldsymbol{S}\langle\alpha{:=}N{\cdot}\gamma\rangle)\,N & (M \neq \mu\beta.\mathrm{C}) \\
(n\boldsymbol{S}): & (\mu\delta.[\alpha]M)\,\boldsymbol{S} & \to & \mu\delta.[\alpha]M\boldsymbol{S} & (\alpha \notin \boldsymbol{S},\ M \neq \mu\beta.\mathrm{C}) \\
(gc): & M\boldsymbol{S} & \to & M\boldsymbol{S}\backslash c & (c \in \boldsymbol{S},\ c \notin M)
\end{array}$$

iv) We only allow the following (unnamed) contextual rules:

$$M \to N \ \Rightarrow\ \begin{cases} \lambda x.M & \to\ \lambda x.N \\ ML & \to\ NL \\ \mu\alpha.[\beta]M & \to\ \mu\alpha.[\beta]N \quad (\alpha \neq \beta \vee \alpha \in M,\ M \neq \mu\delta.\mathrm{C}) \\ M\boldsymbol{S} & \to\ N\boldsymbol{S} \end{cases}$$

Notice that, for example, the substitution in $(\mu\alpha.[\beta]P)\,\vec{Q}\,\langle\beta{:=}N{\cdot}\gamma\rangle$ does not get activated through the *hn*-rule until all leading head R-redexes in $\mu\alpha.[\beta]P$ have been contracted.

Remark 5.2 *i*) It might seem reasonable to allow the propagation inside an abstraction only if the variable concerned is the head-variable of the body, as expressed by:

$$(\lambda y.M)\,\langle x{:=}N\rangle \ \to\ \lambda y.(M\,\langle x{:=}N\rangle) \quad (x = hv(M))$$

but that would imply that a reduction like

$$(\lambda xy.yx)\,QR \ \to_{\mathrm{XH}}\ (\lambda y.yx)\,\langle x{:=}Q\rangle\,R$$

would stop at the last term, since $(\lambda y.yx)\,\langle x{:=}Q\rangle$ is not an abstraction; because we allow the substitution to propagate, we obtain:

$$\begin{array}{lll}
(\lambda xy.yx)\,QR & \to_{\mathrm{XH}}\ ((\lambda y.yx)\,\langle x{:=}Q\rangle)\,R & \to_{\mathrm{XH}}\ \lambda y.(yx\,\langle x{:=}Q\rangle)\,R \\
 & \to_{\mathrm{XH}}\ (yx)\,\langle x{:=}Q\rangle\,\langle y{:=}R\rangle & \to_{\mathrm{XH}}\ Rx\,\langle x{:=}Q\rangle\,\langle y{:=}R\rangle \\
 & \to_{\mathrm{XH}}\ Rx\,\langle x{:=}Q\rangle &
\end{array}$$

We will see below that this is exactly the reduction that our interpretation into the π-calculus represents.

ii) We could have allowed for substitution to propagate also through applications, by defining

$$\begin{array}{rrcl}
(hv): & x\boldsymbol{S}\langle x{:=}N\rangle & \to & N\boldsymbol{S}\langle x{:=}N\rangle \\
(@\boldsymbol{S}): & (MN)\,\boldsymbol{S} & \to & (M\boldsymbol{S})\,(N\boldsymbol{S})
\end{array}$$

Then the substitutions would percolate through the applicative structure and only get activated when reaching the head variable of a term, or a context switch, as in '$\to_{\mathrm{X}}$'. The substitutions in $N\boldsymbol{S}$ inside $(M\boldsymbol{S})\,(N\boldsymbol{S})$ are inactive; therefore we have (assume $x \notin \boldsymbol{S}$):

$$\begin{array}{lll}
(xM(NP))\,\boldsymbol{S} \ \to_{\mathrm{XH}}\ (@\boldsymbol{S}) & ((xM)\,\boldsymbol{S})\,((NP)\,\boldsymbol{S}) & \to_{\mathrm{XH}}\ (@\boldsymbol{S}) \\
 & ((x\boldsymbol{S})\,(M\boldsymbol{S}))\,((NP)\,\boldsymbol{S}) & \to^{*}_{\mathrm{XH}}\ (gc)\quad x\,(M\boldsymbol{S})\,((NP)\,\boldsymbol{S})
\end{array}$$

We have decided against that mainly for reasons of elegance and a better tractable reduction relation, and to get a definition that is closer to Krivine's machine. Notice that, of course, $x\,(M\boldsymbol{S})\,((NP)\,\boldsymbol{S})$ is a correct term in $\lambda\mu\mathbf{x}$.

[15] By Barendregt's convention, none of the free variables and names in N occur in any $\boldsymbol{S}_i$; each $\boldsymbol{S}_i$ can be empty.

$$
\begin{array}{ll}
(\mu\alpha.[\alpha]\lambda y.y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x))(\lambda z.z) & \\
\quad \to_{\text{XH}} (\mu) & \mu\gamma.([\alpha]\lambda y.y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x))\langle\alpha:=\lambda z.z\cdot\gamma\rangle \\
\quad \to_{\text{XH}} (hn) & \mu\gamma.[\gamma]\ (\lambda y.y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x))\langle\alpha:=\lambda z.z\cdot\gamma\rangle\ (\lambda z.z) \\
\quad \to_{\text{XH}} (\lambda\boldsymbol{S}) & \mu\gamma.[\gamma]\ \lambda y.\ (y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x)\langle\alpha:=\lambda z.z\cdot\gamma\rangle)\ (\lambda z.z) \\
\quad \to_{\text{XH}} (\beta) & \mu\gamma.[\gamma]\ y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x)\langle\alpha:=\lambda z.z\cdot\gamma\rangle\langle y:=\lambda z.z\rangle \\
\quad \to_{\text{XH}} (hv) & \mu\gamma.[\gamma]\ (\lambda q.q)(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x)\langle\alpha:=\lambda z.z\cdot\gamma\rangle\langle y:=\lambda z.z\rangle \\
\quad \to_{\text{XH}} (gc) & \mu\gamma.[\gamma]\ (\lambda q.q)(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x)\langle\alpha:=\lambda z.z\cdot\gamma\rangle \\
\quad \to_{\text{XH}} (\beta) & \mu\gamma.[\gamma]\ q\langle q:=\mu\delta.[\alpha]\lambda x.x\rangle(\mu\delta'.[\alpha]\lambda x.x)\langle\alpha:=\lambda z.z\cdot\gamma\rangle \\
\quad \to_{\text{XH}} (hv) & \mu\gamma.[\gamma]\ (\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x)\langle\alpha:=\lambda z.z\cdot\gamma\rangle \\
\quad \to_{\text{XH}} (\mu) & \mu\gamma.[\gamma]\ \mu\delta.([\alpha]\lambda x.x)\langle\delta:=\mu\delta'.[\alpha]\lambda x.x\cdot\gamma'\rangle\langle\alpha:=\lambda z.z\cdot\gamma\rangle \\
\quad \to_{\text{XH}} (gc) & \mu\gamma.[\gamma]\ \mu\delta.[\alpha]\lambda x.x\langle\alpha:=\lambda z.z\cdot\gamma\rangle \\
\quad \to_{\text{XH}} (hn) & \mu\gamma.[\gamma]\ \mu\delta.[\gamma]\ \lambda x.x\langle\alpha:=\lambda z.z\cdot\gamma\rangle(\lambda z.z) \\
\quad \to_{\text{XH}} (gc) & \mu\gamma.[\gamma]\ \mu\delta.[\gamma]\ (\lambda x.x)\ (\lambda z.z) \\
\quad \to_{\text{XH}} (R) & \mu\gamma.[\gamma]\ (\lambda x.x)\ (\lambda z.z) \\
\quad \to_{\text{XH}} (C) & (\lambda x.x)(\lambda z.z)\ \to_{\text{XH}} (\beta)\ x\langle x:=\lambda z.z\rangle\ \to_{\text{XH}} (hv)\ \ \lambda z.z
\end{array}
$$

Figure 2: Running $(\mu\alpha.[\alpha]\lambda y.y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x))(\lambda z.z)$ in '$\to_{\text{XH}}$'.

Definition 5.3 (Head normal forms for $\lambda\mu\mathbf{x}$) The head forms with respect to '$\to_{\text{x}}$' are defined through the grammar:

$$
\begin{array}{rcll}
\boldsymbol{N} & ::= & \lambda x.\boldsymbol{N} & \\
 & \mid & xM_1\boldsymbol{S}_1\cdots M_n\boldsymbol{S}_n & (n\geq 0, \forall i\in \underline{n}\ (x\notin \boldsymbol{S}_i)) \\
 & \mid & \mu\alpha.[\beta]\boldsymbol{N} & (\alpha\neq\beta\vee\alpha\in\boldsymbol{N},\ \boldsymbol{N}\neq\mu\gamma.[\delta]\boldsymbol{N}')
\end{array}
$$

It is straightforward to check that these terms are normal forms with respect to '$\to_{\text{XH}}$'.

The next proposition states the relation between explicit head-reduction, head reduction, and explicit reduction.

Lemma 5.4 *i)* $M\langle x:=N\rangle \to^*_{:=} M\{N/x\}$.

ii) $M\langle\alpha:=N\cdot\gamma\rangle \to^*_{:=} M\{N\cdot\gamma/\alpha\}$.

iii) *If* $M\to^*_{\text{H}} N$, *then there exists* $L\in\lambda\mu\mathbf{x}$ *such that* $M\to^*_{\text{XH}} L$ *and* $L\to^*_{:=} N$.

iv) *If* $P\to^*_{\text{XH}} Q$ *then there exists* $R,S\in\lambda\mu$ *such that* $P\to^{nf}_{:=} R$, *and* $Q\to^{nf}_{:=} S$, *and* $R\to^*_{\text{H}} S$.

v) *If* $P\to^{nf}_{\text{XH}} Q$ *with* $P\in\lambda\mu$, *then there exists* $R\in\lambda\mu$ *such that* $Q\to^{nf}_{:=} R$, *and* $P\to^{nf}_{\text{H}} R$.

vi) $M\to^{nf}_{\beta\mu} N$ *if and only if there exists* $L\in\lambda\mu\mathbf{x}$ *such that* $M\to^{nf}_{\text{XH}} L$ *and* $L\to^{nf}_{\text{x}} N$.

Proof: The first two parts are straightforward by induction on the structure of terms. For the third, the proof is by straightforward induction on the number of reduction steps. For the fourth, all cases are straightforward, if not trivial; for example, we have

(hv): Then $P = x\boldsymbol{S}_0M_1\boldsymbol{S}_1\cdots M_n\boldsymbol{S}_n\langle x:=N\rangle$, and $Q = N\boldsymbol{S}_0M_1\boldsymbol{S}_1\cdots M_n\boldsymbol{S}_n\langle x:=N\rangle$.
Let $x\boldsymbol{S}_0M_1\boldsymbol{S}_1\cdots M_n\boldsymbol{S}_n\langle x:=N\rangle \to^{nf}_{:=} NM'_1\cdots M'_n = R = S$.

(hn): Then $P = (\mu\delta.[\alpha]M)\boldsymbol{S}\langle\alpha:=N\cdot\gamma\rangle$, and $Q = \mu\delta.[\gamma](M\boldsymbol{S}\langle\alpha:=N\cdot\gamma\rangle)N$. Let $M\boldsymbol{S}\langle\alpha:=N\cdot\gamma\rangle \to^{nf}_{:=} M'$ and $N\to^{nf}_{:=} N'$, then $\mu\delta.[\gamma](M\boldsymbol{S}\langle\alpha:=N\cdot\gamma\rangle)N \to^{nf}_{:=} \mu\delta.[\gamma]M'N'$; take $R = \mu\delta.[\gamma]M'N' = S$.

The fifth is a special case of the fourth, and the sixth follows easily. □

This result gives that we can show our main results for $\lambda\mu\mathbf{x}$ for reductions that reduce to head-normal form.

We give some examples that illustrate $\lambda\mu\mathbf{x}$ and '$\to_{\text{XH}}$'.

Example 5.5 *i)* As an example where the special character of explicit head-reduction for $\lambda\mu$ becomes more clear, take the reduction of $(\mu\alpha.[\alpha]\lambda y.y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x))(\lambda z.z)$ in Fig. 2. We will see in Fig. 3 how this reduction is modelled in our interpretation.

ii) Reduction in '$\to_{\text{XH}}$' is not deterministic in general:

$$
(\lambda x.(\lambda y.M)N)L\ \to_{\text{XH}}\ \begin{cases}(\lambda x.M\langle y:=N\rangle)L\\ ((\lambda y.M)N)\langle x:=L\rangle\end{cases}
$$

Since both these reductions are respected under our interpretation, we will not give one priority over the other.

iii) Of course in '$\to_{\text{XH}}$' we can have non-terminating reductions. We know that in '$\to_{\beta\mu}$' and '$\to_{\text{H}}$', $(\lambda x.xx)\,(\lambda x.xx)$ reduces to itself; this is not the case for '$\to_{\text{XH}}$', as is illustrated by (where $\Delta = \lambda x.xx$):

$$\begin{array}{lllll}
\Delta\Delta & \triangleq & (\lambda x.xx)\,\Delta & \to_{\text{XH}} & xx\,\langle x{:=}\Delta\rangle \\
 & \to_{\text{XH}} & (\lambda y.yy)\,x\,\langle x{:=}\Delta\rangle & \to_{\text{XH}} & yy\,\langle y{:=}x\rangle\,\langle x{:=}\Delta\rangle \\
 & \to_{\text{XH}} & xy\,\langle y{:=}x\rangle\,\langle x{:=}\Delta\rangle & \to_{\text{XH}} & (\lambda z.zz)y\,\langle y{:=}x\rangle\,\langle x{:=}\Delta\rangle \\
 & \to_{\text{XH}} & zz\,\langle z{:=}y\rangle\,\langle y{:=}x\rangle\,\langle x{:=}\Delta\rangle & \to^*_{\text{XH}} & \cdots
\end{array}$$

(notice the α-conversions, needed to adhere to Barendregt's convention). This reduction is deterministic and clearly does not terminate. Notice that $\Delta\Delta$ does not run to itself; however,

$$\begin{array}{lll}
zz\,\langle z{:=}y\rangle\,\langle y{:=}x\rangle\,\langle x{:=}\Delta\rangle & \to^*_{:=} & yy\,\langle y{:=}x\rangle\,\langle x{:=}\Delta\rangle \\
 & \to^*_{:=} & xx\,\langle x{:=}\Delta\rangle \\
 & \to^*_{:=} & \Delta\Delta
\end{array}$$

so, as stated by Lemma 5.4, the standard reduction result can be achieved by reduction in '$\to_{:=}$' (we will use Δ for $\lambda x.xx$ again below).

6 A logical interpretation of $\lambda\mu$x-terms to π-processes

We will now define our logical, output-based interpretation $\llbracket M \rrbracket^{\text{L}}_{a}$ of the $\lambda\mu$**x**-calculus into the π-calculus (where M is a $\lambda\mu$**x**-term, and a is the name given to its (anonymous) output), which is essentially the one presented in [13], but no longer considers $[\alpha]M$ to be a term.

The main idea behind the interpretation, as in [12], is to give a name to the anonymous output of terms; it combines this with the inherent naming mechanism of $\lambda\mu$. As we will show in Thm 7.1, this encoding naturally represents explicit head-reduction; we will need to consider weak reduction later for the full abstraction result, but not for soundness, completeness, or termination.

The interpretation of $\lambda\mu$**x** terms into the π-calculus is defined by:

Definition 6.1 (Logical interpretation of $\lambda\mu$**x** terms (*cf.* [13])) Let a not be a $\lambda\mu$-variable or name. Then

$$\begin{array}{rcll}
\llbracket x \rrbracket^{\text{L}}_{a} & \triangleq & x(u).!u{\to}\overline{a} & (u \textit{ fresh}) \\
\llbracket \lambda x.M \rrbracket^{\text{L}}_{a} & \triangleq & (\nu xb)\,(\llbracket M \rrbracket^{\text{L}}_{b} \mid \overline{a}\langle x,b\rangle) & (b \textit{ fresh}) \\
\llbracket MN \rrbracket^{\text{L}}_{a} & \triangleq & (\nu c)\,(\llbracket M \rrbracket^{\text{L}}_{c} \mid !c(v,d).(\llbracket v{:=}N \rrbracket^{\text{L}}_{} \mid !d{\to}\overline{a})) & (c,v,d \textit{ fresh}) \\
\llbracket M\langle x{:=}N\rangle \rrbracket^{\text{L}}_{a} & \triangleq & (\nu x)\,(\llbracket M \rrbracket^{\text{L}}_{a} \mid \llbracket x{:=}N \rrbracket^{\text{L}}_{}) & \\
\llbracket x := N \rrbracket^{\text{L}}_{} & \triangleq & !\overline{x}(w).\llbracket N \rrbracket^{\text{L}}_{} w & (w \textit{ fresh}) \\
\llbracket \mu\gamma.[\beta]M \rrbracket^{\text{L}}_{a} & \triangleq & \llbracket M \rrbracket^{\text{L}}_{} \beta\,\{a/\gamma\} & \\
\llbracket M\langle \beta{:=}N{\cdot}\gamma\rangle \rrbracket^{\text{L}}_{a} & \triangleq & (\nu\beta)\,(\llbracket M \rrbracket^{\text{L}}_{a} \mid \llbracket \beta{:=}N{\cdot}\gamma \rrbracket^{\text{L}}_{}) & \\
\llbracket \alpha := N{\cdot}\gamma \rrbracket^{\text{L}}_{} & \triangleq & !\alpha(v,d).(\llbracket v{:=}N \rrbracket^{\text{L}}_{} \mid !d{\to}\overline{\gamma}) & (v,d \textit{ fresh})
\end{array}$$

Notice that this definition uses forwarders (see Def. 2.1:(*vii*)).

The interpretation of $\mu\gamma.[\beta]M$ is in fact a combination of two alternatives of the encoding presented in [13]:

$$\begin{array}{rcll}
\llbracket \mu\gamma.\text{C} \rrbracket^{\text{L}}_{a} & \triangleq & (\nu s)\,\llbracket \text{C} \rrbracket^{\text{L}}_{} s\,\{a/\gamma\} & (s \textit{ fresh}) \\
\llbracket [\beta]M \rrbracket^{\text{L}}_{a} & \triangleq & \llbracket M \rrbracket^{\text{L}}_{} \beta &
\end{array}$$

Remark 6.2 We can make the following observations:

- Explicit substitution $\langle x{:=}N\rangle$ is encoded through replication; as we will see below in the proof of Thm 7.1, each individual occurrence of (the encoding of) a variable gets treated on its own, so replication is needed to guarantee that the execution of substitution does not deplete the source. We block the running of the encoding of N by placing it under an output guard: we interpret each 'incarnation' of the encoding of N under a new name w, and send that name out to that occurrence of the encoding of x that N should be substituted for. This implies that a variable x is interpreted as a process that first receives the name under which the encoding of N outputs, and then uses that name to establish the redirection.
- For an abstraction $\lambda x.M$, we give the name b to the output of M; that M has input x and output b gets sent out over a, which is the name of $\lambda x.M$, so that a process that wants to call on this functionality, knows which channel to send the input to, and on which channel to pick up the result.[16]
- For the interpretation of an abstraction $(\nu xb)\,(\llbracket M\rrbracket^{\mathrm{L}}_{a}\, b \mid \overline{a}\langle x,b\rangle)$, the output over a of the channel names x and b is placed in parallel to the interpretation of M under b, and can communicate asynchronously. We cannot restrict the co-domain of our interpretation to the asynchronous π-calculus, however, since to achieve completeness an output guard is needed for the interpretation of an explicit substitution.
- For an application MN, the pair of the names of the (first) input and output channels of M, transmitted over c, is received as a pair $\langle v,d\rangle$ of input-output names in the right-hand side; the received input v name is used to send the output name for the encoding of N, enabling the simulation of substitution, and the received output name d gets redirected to the output of the application a. Since a name α can appear many times in M, when we interpret $\llbracket(\mu\alpha.[\beta]M)N\rrbracket^{\mathrm{L}}_{a} \triangleq (\nu c)\,(\llbracket M\rrbracket^{\mathrm{L}}_{c}\,\beta\,\{c/\alpha\} \mid !c(v,d).(\llbracket v{:=}N\rrbracket^{\mathrm{L}}_{d} \mid !d{\rightarrow}\overline{a}))$ we need to be able to deal with the multiple outputs over c in $\llbracket M\rrbracket^{\mathrm{L}}_{c}\,\beta\,\{c/\alpha\}$, so the part that deals with the input over c has to be replicated.
- For the context switch $\mu\gamma.[\beta]M$, we use the fact that the name β is the name given to M in $\lambda\mu\mathbf{x}$, and use that name for the main output of the interpretation of M. The operands for $\mu\gamma.[\beta]M$ are in fact for the terms named γ; since the output name we give to the process is a, where a context might seek to communicate with, we need to rename all the occurrences of γ in the interpreted process by a.
- Since $\llbracket \boldsymbol{S}_{c}\rrbracket^{\mathrm{L}}$ is either $!c(v,d).(\llbracket v{:=}N\rrbracket^{\mathrm{L}}_{d} \mid !d{\rightarrow}\overline{\gamma})$ or $!\overline{c}(w).\llbracket N\rrbracket^{\mathrm{L}}_{w}\, w$, the process $\llbracket \boldsymbol{S}_{c}\rrbracket^{\mathrm{L}}$ can only input or only output on c, so $(\nu c)\,(\llbracket \boldsymbol{S}_{c}\rrbracket^{\mathrm{L}}) \approx 0$.

The interpretation is called logical since the structure of the encoding of application corresponds to how Gentzen translates the *modus ponens* inference rule of natural deduction (on the left) in the sequent calculus [30] (on the right); see Thm 4.8 in [11]. :

$$\dfrac{\boxed{\phantom{\Gamma \vdash_{\mathrm{ND}} A{\rightarrow}B}} \qquad \boxed{\phantom{\Gamma \vdash_{\mathrm{ND}} A}}}{\Gamma \vdash_{\mathrm{ND}} B}\,(\rightarrow E) \qquad\qquad \dfrac{\dfrac{\boxed{\phantom{\Gamma \vdash_{\mathrm{LK}} A{\rightarrow}B}}}{\Gamma \vdash_{\mathrm{LK}} A{\rightarrow}B, B}\,(Weak) \qquad \dfrac{\boxed{\phantom{\Gamma \vdash_{\mathrm{LK}} A}} \qquad \dfrac{}{\Gamma, B \vdash_{\mathrm{LK}} B}\,(Ax)}{\Gamma, A{\rightarrow}B \vdash_{\mathrm{LK}} B}\,(\rightarrow L)}{\Gamma \vdash_{\mathrm{LK}} B}\,(cut)$$

Remark 6.3 *i*) As mentioned above, the interpretation presented in [13] had the case

$$\llbracket \mu\gamma.M\rrbracket^{\mathrm{L}}_{a} \;\triangleq\; (\nu s)\,\llbracket M\rrbracket^{\mathrm{L}}_{s}\,\{a/\gamma\} \quad (s\ \textit{fresh})$$

so was defined for $\Lambda\mu$ (notice the use of M rather than C). Note that this encoding elegantly expresses that the main computation in $\mu\gamma.M$ is blocked: the name s is fresh and bound and never transmitted, so the main output of $(\nu s)\,\llbracket M\rrbracket^{\mathrm{L}}_{s}\,\{a/\gamma\}$ cannot be received.

[16] This view of computation is exactly that of the calculus $\mathcal{X}$.

However, in order to achieve full abstraction, we had to restrict our interpretation to $\lambda\mu$, so no longer can consider $[\alpha]M$ a term. The reason is that, using that interpretation, the process

$$\llbracket \mu\alpha.\lambda x.x \rrbracket^{\mathtt{L}} a \;\triangleq\; (\nu s)\,((\nu xb)\,(x(u).!u{\rightarrow}\overline{b} \mid \overline{s}\langle x,b\rangle))$$

is in normal form. Notice that all inputs and outputs are restricted; thereby, this process is weakly bisimilar to $\mathit{0}$ and to $\llbracket \Delta\Delta \rrbracket^{\mathtt{L}} a$ (see Lemma 8.7). So using that interpretation, we cannot distinguish between *blocked* and *looping* computations, which clearly would affect any full-abstraction result. When restricting our interpretation to $\lambda\mu$, this problem disappears: since naming has to follow μ-abstraction, $\mu\alpha.[\lambda x.x]$is not a term in $\lambda\mu$; instead, now (assuming $\alpha \neq \beta$)

$$\llbracket \mu\alpha.[\beta]\lambda x.x \rrbracket^{\mathtt{L}} a \;\triangleq\; (\nu xb)\,(\llbracket x \rrbracket^{\mathtt{L}} b \mid \overline{\beta}\langle x,b\rangle)\,\{a/\alpha\} \;\triangleq\; (\nu xb)\,(\llbracket x \rrbracket^{\mathtt{L}} b \mid \overline{\beta}\langle x,b\rangle)$$

which outputs on β, so is not weakly bisimilar to $\mathit{0}$.

ii) Note that we could have avoided the implicit renaming in the case for μ-abstraction by defining

$$\llbracket \mu\gamma.[\delta]M \rrbracket^{\mathtt{L}} a \;\triangleq\; (\nu\gamma)\,(\llbracket M \rrbracket^{\mathtt{L}} \delta \mid !\gamma{\rightarrow}\overline{a})$$

which is operationally the same as $\llbracket M \rrbracket^{\mathtt{L}} \delta\,\{a/\gamma\}$ (they are, in fact, weakly bisimilar) but then we could not show that terms in $\rightarrow_{\mathrm{XH}}$-normal form are translated to processes in normal form (Lemma 7.8), a property that is of use in the proof of termination (Thm 7.9).

iii) To underline the significance of our results, notice that the encoding is not trivial (so does not equate all terms), since

$$\begin{aligned}\llbracket \lambda yz.y \rrbracket^{\mathtt{L}} a &\;\triangleq\; (\nu yb)\,((\nu zd)\,(y(u).!u{\rightarrow}\overline{d} \mid \overline{b}\langle z,d\rangle) \mid \overline{a}\langle y,b\rangle)\\ \llbracket \lambda x.x \rrbracket^{\mathtt{L}} a &\;\triangleq\; (\nu xb)\,(x(u).!u{\rightarrow}\overline{b} \mid \overline{a}\langle x,b\rangle)\end{aligned}$$

processes that differ under '$\approx$': notice that the first exhibits two outputs so can interact twice with a context providing two inputs, whereas the second only exhibits one and cannot interact twice, so can be distinguished.

iv) Notice that, as is the case for Milner's interpretation and in contrast to the spine interpretation of [12], a guard is placed on the replicated terms. This is not only done with an eye on proving completeness or preservation of termination, but more importantly, to make sure that $(\nu x)\,(\llbracket x{:=}N \rrbracket^{\mathtt{L}}) \approx \mathit{0}$, a property we need for our full abstraction result: since a term can have named sub-terms, the interpretation will generate output not only for the term itself, but also for those named terms, so the process $(\nu x)\,(!\llbracket N \rrbracket^{\mathtt{L}} x)$ – using the variant of [12] – *can have* observable behaviour, in contrast to here, where $(\nu x)\,(!\overline{x}(w).\llbracket N \rrbracket^{\mathtt{L}} w)$ *is* weakly bisimilar to $\mathit{0}$. Another advantage is that now it is impossible for unintended synchronisations between interpretations of explicit substitutions to take place, a property we need for Thm 7.5 and 12.1.

We illustrate the encoding.

Example 6.4 *i*) We have:

$$\begin{aligned}&\llbracket (zz)\,\langle z{:=}(y\,\langle y{:=}(x\,\langle x{:=}\Delta\rangle)\rangle)\rangle \rrbracket^{\mathtt{L}} a \;\triangleq\; (\nu z)\,((\nu c)\,(z(u).!u{\rightarrow}\overline{c} \mid\\ &\qquad !c(v,d).(\llbracket v{:=}z \rrbracket^{\mathtt{L}} \mid !d{\rightarrow}\overline{a})) \mid !\overline{z}(w).(\nu v)\,(\llbracket y \rrbracket^{\mathtt{L}} v \mid !y(w).(\nu x)\,(\llbracket x \rrbracket^{\mathtt{L}} w \mid !\overline{x}(w).\llbracket \Delta \rrbracket^{\mathtt{L}} w) \mid v{\rightarrow}\overline{w}))\\ &\llbracket zz\,\langle z{:=}y\rangle\,\langle y{:=}x\rangle\,\langle x{:=}\Delta\rangle \rrbracket^{\mathtt{L}} a \;\triangleq\; (\nu x)\,((\nu y)\,((\nu z)\,((\nu c)\,(z(u).!u{\rightarrow}\overline{c} \mid\\ &\qquad\qquad !c(v,d).(\llbracket v{:=}z \rrbracket^{\mathtt{L}} \mid !d{\rightarrow}\overline{a})) \mid !\overline{z}(w).\llbracket y \rrbracket^{\mathtt{L}} w) \mid !\overline{y}(w).\llbracket x \rrbracket^{\mathtt{L}} w) \mid !\overline{x}(w).\llbracket \Delta \rrbracket^{\mathtt{L}} w)\end{aligned}$$

$$
\begin{array}{rl}
& \llbracket(\mu\alpha.[\alpha]\lambda y.y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x))(\lambda z.z)\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, a \\
\triangleq & (\nu c)\,(\llbracket\mu\alpha.[\alpha]\lambda y.y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x)\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, c \mid \llbracket c := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
\triangleq & (\nu c)\,(\llbracket\lambda y.y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x)\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, \alpha\,\{c/\alpha\} \mid \llbracket c := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
=_{\alpha} & (\nu\alpha)\,(\llbracket\lambda y.y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x)\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, \alpha \mid \llbracket \alpha := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
\triangleq & (\nu\alpha)\,((\nu yb)\,(\llbracket y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x)\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, b \mid \overline{\alpha}\langle y,b\rangle) \mid \llbracket \alpha := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
\triangleq & (\nu\alpha)\,((\nu yb)\,((\nu c')\,(\llbracket y(\mu\delta.[\alpha]\lambda x.x)\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, c' \mid \llbracket c' := (\mu\delta'.[\alpha]\lambda x.x)\cdot b\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid \overline{\alpha}\langle y,b\rangle) \mid \llbracket \alpha := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
\triangleq & (\nu\alpha)\,((\nu yb)\,((\nu c')\,((\nu c'')\,(\llbracket y\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, c'' \mid \llbracket c'' := (\mu\delta.[\alpha]\lambda x.x)\cdot c'\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid \\
& \qquad \llbracket c' := (\mu\delta'.[\alpha]\lambda x.x)\cdot b\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid \overline{\alpha}\langle y,b\rangle) \mid \llbracket \alpha := \lambda z.z\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
\equiv & (\nu\alpha)\,((\nu yb)\,((\nu c')\,((\nu c'')\,(\llbracket y\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, c'' \mid \llbracket c'' := (\mu\delta.[\alpha]\lambda x.x)\cdot c'\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid \\
& \qquad \llbracket c' := (\mu\delta'.[\alpha]\lambda x.x)\cdot b\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid \overline{\alpha}\langle y,b\rangle) \mid \alpha(v,d).(\llbracket v := \lambda z.z\rrbracket^{\mathrm{L}}_{\mathbb{I}} \mid !d\rightarrow\overline{a}) \mid \llbracket \alpha := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
(*)\ \triangleq,\rightarrow_{\pi}(\alpha) & (\nu\alpha)\,((\nu yb)\,((\nu c')\,((\nu c'')\,(y(u).!u\rightarrow\overline{c''} \mid \llbracket c'' := (\mu\delta.[\alpha]\lambda x.x)\cdot c'\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid \\
& \qquad \llbracket c' := (\mu\delta.[\alpha]\lambda x.x)\cdot b\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid \overline{y}(w).\llbracket\lambda z.z\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, w \mid !b\rightarrow\overline{a}) \mid \llbracket \alpha := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
(hv)\ \rightarrow_{\pi}(y) & (\nu\alpha)\,((\nu yb)\,((\nu c')\,((\nu c'')\,((\nu w)\,(\llbracket\lambda q.q\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, w \mid !w\rightarrow\overline{c''}) \mid \llbracket c'' := (\mu\delta'.[\alpha]\lambda x.x)\cdot c'\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid \\
& \qquad \llbracket c' := (\mu\delta'.[\alpha]\lambda x.x)\cdot b\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid !b\rightarrow\overline{a}) \mid \llbracket \alpha := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
\triangleq & (\nu\alpha)\,((\nu yb)\,((\nu c')\,((\nu c'')\,((\nu w)\,((\nu qb')\,(\llbracket q\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, b' \mid \overline{w}\langle q,b'\rangle) \mid !w\rightarrow\overline{c''}) \mid \\
& \qquad !c''(v,d).(\llbracket v := \mu\delta.[\alpha]\lambda x.x\rrbracket^{\mathrm{L}}_{\mathbb{I}} \mid !d\rightarrow\overline{c'})) \mid \llbracket c' := (\mu\delta'.[\alpha]\lambda x.x)\cdot b\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid !b\rightarrow\overline{a}) \mid \llbracket \alpha := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
\approx,\triangleq,\rightarrow_{\pi}(w,c'') & (\nu\alpha)\,((\nu yb)\,((\nu c')\,((\nu qb')\,(q(u).!u\rightarrow\overline{b'} \mid !\overline{q}(w).\llbracket\mu\delta.[\alpha]\lambda x.x\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, w \mid !b'\rightarrow\overline{c'}) \mid \\
& \qquad \llbracket c' := (\mu\delta'.[\alpha]\lambda x.x)\cdot b\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid !b\rightarrow\overline{a}) \mid \llbracket \alpha := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
(hv)\ \approx,\rightarrow_{\pi}(q) & (\nu\alpha)\,((\nu yb)\,((\nu c')\,((\nu wb')\,(!w\rightarrow\overline{b'} \mid \llbracket\mu\delta.[\alpha]\lambda x.x\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, w \mid !b'\rightarrow\overline{c'}) \mid \\
& \qquad \llbracket c' := (\mu\delta'.[\alpha]\lambda x.x)\cdot b\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid !b\rightarrow\overline{a}) \mid \llbracket \alpha := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
(hn) & \triangleq (\nu\alpha)\,((\nu yb)\,((\nu c')\,((\nu wb')\,(!w\rightarrow\overline{b'} \mid \llbracket\lambda x.x\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, \alpha\,\{w/\delta\} \mid !b'\rightarrow\overline{c'}) \mid \\
& \qquad \llbracket c' := (\mu\delta'.[\alpha]\lambda x.x)\cdot b\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid !b\rightarrow\overline{a}) \mid \llbracket \alpha := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
\triangleq & (\nu\alpha)\,((\nu yb)\,((\nu c')\,((\nu wb')\,(!w\rightarrow\overline{b'} \mid (\nu xb'')\,(\llbracket x\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, b'' \mid \overline{\alpha}\langle x,b''\rangle) \mid !b'\rightarrow\overline{c'}) \mid \\
& \qquad \llbracket c' := (\mu\delta'.[\alpha]\lambda x.x)\cdot b\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \mid !b\rightarrow\overline{a}) \mid \llbracket \alpha := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
\approx,\triangleq & (\nu\alpha)\,((\nu xb'')\,(\llbracket x\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, b'' \mid \overline{\alpha}\langle x,b''\rangle) \mid !\alpha(v,d).(\llbracket v := \lambda z.z\rrbracket^{\mathrm{L}}_{\mathbb{I}} \mid !d\rightarrow\overline{a})) \\
(*)\ \triangleq,\rightarrow_{\pi}(\alpha) & (\nu xb'')\,(x(u).!u\rightarrow\overline{b''} \mid \overline{x}(w).(\nu zb)\,(\llbracket z\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, b \mid \overline{w}\langle z,b\rangle) \mid !b''\rightarrow\overline{a}) \mid (\nu\alpha)\,(\llbracket \alpha := (\lambda z.z)\cdot a\rrbracket^{\mathrm{L}}_{\mathbb{I}}) \\
(hv)\ \approx,\rightarrow_{\pi}(x,w) & (\nu zb'')\,(\llbracket z\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, b'' \mid \overline{a}\langle z,b''\rangle) \quad \triangleq \llbracket\lambda z.z\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, a
\end{array}
$$

Figure 3: Running $\llbracket(\mu\alpha.[\alpha]\lambda y.y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x))(\lambda z.z)\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, a$ in '$\rightarrow_{\pi}$'.

$$
\begin{array}{rl}
ii)\ \llbracket(\lambda x.MN)P\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, a & \triangleq (\nu c)\,(\llbracket\lambda x.MN\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, c \mid !c(v,d).(\llbracket v := P\rrbracket^{\mathrm{L}}_{\mathbb{I}} \mid !d\rightarrow\overline{a})) \\
& \triangleq (\nu c)\,((\nu xb)\,(\llbracket MN\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, b \mid \overline{c}\langle x,b\rangle) \mid !c(v,d).(\llbracket v := P\rrbracket^{\mathrm{L}}_{\mathbb{I}} \mid !d\rightarrow\overline{a})) \\
& \triangleq (\nu c)\,((\nu xb)\,(\llbracket MN\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, b \mid \overline{c}\langle x,b\rangle) \mid !c(v,d).(\llbracket v := P\rrbracket^{\mathrm{L}}_{\mathbb{I}} \mid !d\rightarrow\overline{a})) \\
\rightarrow_{\pi}(c) & (\nu xb)\,((\nu c')\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, c' \mid !c'(v,d).(\llbracket v := N\rrbracket^{\mathrm{L}}_{\mathbb{I}} \mid !d\rightarrow\overline{b})) \mid \\
& \qquad \llbracket x := P\rrbracket^{\mathrm{L}}_{\mathbb{I}} \mid !b\rightarrow\overline{a}) \mid (\nu c)\,(!c(v,d).(\llbracket v := P\rrbracket^{\mathrm{L}}_{\mathbb{I}} \mid !d\rightarrow\overline{a})) \\
& \approx (\nu x)\,((\nu c')\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, c' \mid !c'(v,d).(\llbracket v := N\rrbracket^{\mathrm{L}}_{\mathbb{I}} \mid !d\rightarrow\overline{a})) \mid !\overline{x}(w).\llbracket P\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, w) \\
& \triangleq \llbracket MN\langle x := P\rangle\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, a
\end{array}
$$

Assume x does not occur in N, then:

$$
\begin{array}{rl}
\llbracket MN\langle x := P\rangle\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, a & \triangleq (\nu x)\,((\nu c')\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, c' \mid !c'(v,d).(\llbracket v := N\rrbracket^{\mathrm{L}}_{\mathbb{I}} \mid !d\rightarrow\overline{a})) \mid !\overline{x}(w).\llbracket P\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, w) \\
& \equiv (\nu c')\,((\nu x)\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, c' \mid !\overline{x}(w).\llbracket P\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, w) \mid !c'(v,d).(\llbracket v := N\rrbracket^{\mathrm{L}}_{\mathbb{I}} \mid !d\rightarrow\overline{a})) \\
& \triangleq \llbracket M\langle x := P\rangle N\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, a
\end{array}
$$

This goes to show that, since substitutions are placed in parallel, through congruence we can model unwanted propagation of substitution; notice that $(\lambda x.MN)\,P$ does not run to $M\langle x := P\rangle N$, not even when allowing for the rules as discussed in Rem 5.2. We will see this observation play a role in Thm 7.5.

In [12] the case for application in the interpretation for λ-terms was defined as:

$$
\llbracket MN\rrbracket^{\mathrm{s}}_{\mathbb{I}}\, a \ \triangleq\ (\nu c)\,(\llbracket M\rrbracket^{\mathrm{H}}_{\mathbb{I}}\, c \mid c(v,d).(\llbracket v := N\rrbracket^{\mathrm{s}}_{\mathbb{I}} \mid d\rightarrow\overline{a}))
$$

In particular, there the input on name c is *not replicated*: this corresponds to the fact that for λ-terms, in $\llbracket M\rrbracket^{\mathrm{H}}_{\mathbb{I}}\, c$, the output c is used *exactly once*, which is not the case for the interpretation of $\lambda\mu$-terms: for example, α might appear many times in M, and since $\llbracket\mu\alpha.[\alpha]M\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, c = \llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, \alpha\,\{c/\alpha\} = \llbracket M\{c/\alpha\}\rrbracket^{\mathrm{L}}_{\mathbb{I}}\, c$, then the name c appears many times in the latter.

Remark 6.5 Observe the similarity between

$$\begin{array}{rcl} \llbracket MN \rrbracket^{\mathrm{L}}_{\rrbracket} a & \triangleq & (\nu c)\,(\llbracket M \rrbracket^{\mathrm{L}}_{\rrbracket} c \mid !c(v,d).(\llbracket v := N \rrbracket^{\mathrm{L}}_{\rrbracket} \mid !d{\rightarrow}\overline{a})) \quad \text{and} \\ \llbracket M\langle c := N\cdot\gamma\rangle \rrbracket^{\mathrm{L}}_{\rrbracket} a & \triangleq & (\nu c)\,(\llbracket M \rrbracket^{\mathrm{L}}_{\rrbracket} a \mid !c(v,d).(\llbracket v := N \rrbracket^{\mathrm{L}}_{\rrbracket} \mid !d{\rightarrow}\overline{\gamma})) \end{array}$$

The first communicates N via the main output channel c of M (which might occur more than once inside $\llbracket M \rrbracket^{\mathrm{L}}_{\rrbracket} c$, so replication is needed), whereas the second communicates with all the sub-processes that have c as output name, and changes the output name of the process to γ.[17] In other words, application is just a special case of explicit structural substitution. As an abbreviation, we sometimes will write $(\nu c)\,(\llbracket M \rrbracket^{\mathrm{L}}_{\rrbracket} c \mid \llbracket c := N\cdot a \rrbracket^{\mathrm{L}}_{\rrbracket})$ for $\llbracket MN \rrbracket^{\mathrm{L}}_{\rrbracket} a$.

This observation plays a prominent role in the proof of Thm 7.1, in the case dealing with reduction step (hn). There

$$\llbracket \langle \beta := N\cdot\gamma\rangle \rrbracket^{\mathrm{L}}_{\rrbracket} \;\triangleq\; !\beta(v,d).(\llbracket v := N \rrbracket^{\mathrm{L}}_{\rrbracket} \mid !d{\rightarrow}\overline{\gamma}) \;\in\; \llbracket \boldsymbol{S} \rrbracket^{\mathrm{L}}_{\rrbracket},$$

is used to represent the explicit substitution $\langle \beta := N\cdot\gamma\rangle$ in the interpretation of the contractum. However, in the implementation of this step we should also 'generate' the new occurrence of N that gets placed behind $M\langle \beta := N\cdot\gamma\rangle$ in the new application. This turns out to be straightforward, since that right-hand side of application is also represented by $\llbracket \langle \beta := N\cdot\gamma\rangle \rrbracket^{\mathrm{L}}_{\rrbracket}$, and we can use that $!P \approx !P \mid !P$.

This is illustrated in Fig. 3; as the first step, the contextual substitution

$$\llbracket \alpha := (\lambda z.z)\cdot a \rrbracket^{\mathrm{L}}_{\rrbracket} \;\triangleq\; !\alpha(v,d).(\llbracket v := \lambda z.z \rrbracket^{\mathrm{L}}_{\rrbracket} \mid !d{\rightarrow}\overline{a})$$

gets created directly by definition of $\llbracket \cdot \rrbracket^{\mathrm{L}}_{\rrbracket} \cdot$. This is used twice, in the steps marked $(*)$; the first exchanges the pair $\langle y,b\rangle$ over α which creates the process $\llbracket y := \lambda z.z \rrbracket^{\mathrm{L}}_{\rrbracket} \triangleq !\overline{y}(w).\llbracket \lambda z.z \rrbracket^{\mathrm{L}}_{\rrbracket} w$ so that the substitution of the head variable y by $\lambda z.z =_{\alpha} \lambda q.q$ can be modelled, in the first step marked (hv); here $\llbracket \alpha := (\lambda z.z)\cdot a \rrbracket^{\mathrm{L}}_{\rrbracket}$ acts for the outermost application. The second use is again for a synchronisation over α, but now uses $\llbracket \alpha := (\lambda z.z)\cdot a \rrbracket^{\mathrm{L}}_{\rrbracket}$ to represent the explicit substitution.

Notice that context switches do not really influence the structure of the process that is created by the interpretation since they have no representation in π, but are statically encoded through renaming. And although the notion of structural reduction in $\lambda\mu$ is very different from normal β-reduction, no special measures had to be taken in our encoding in order to be able to express it; the component of our interpretation that deals with pure λ-terms is almost exactly that of [12] (ignoring for the moment that substitution is modelled using a guard, which affects also the interpretation of variables), but for the use of replication in the case for application; we will come back to this in Thm A.8. In fact, the distributive character of structural substitution is dealt with entirely by congruence (see also Example 6.9).

This strengthens our view that, as far as our interpretation is concerned, μ-reduction is not a separate computational step, but essentially is static administration, a reorganisation of the applicative structure of a term, which has to be defined explicitly in the context of the λ-calculus, but is dealt with by our interpretation statically rather than by synchronisation between processes in the π-calculus. In fact, modelling β-reduction in the π-calculus involves a computational step, but context switches are dealt with by congruence; this is only possible, of course, because the interpretation of the operand in application uses replication. This stresses that the π-calculus constitutes a very powerful abstract machine indeed.

We would like to stress that, although inspired by logic, our interpretation does not depend on types *at all*; in fact, we can treat untypeable terms as well, and can show that $\llbracket \Delta\Delta \rrbracket^{\mathrm{L}}_{\rrbracket} a$ (perhaps the prototype of a non-typeable term) runs forever without generating output (see Example 8.1; this already holds for the interpretation of [12]).

[17] A similar observation can be made for the interpretation of $\lambda\mu$ in $\mathcal{X}$ [11].

Remark 6.6 Substitutions are interpreted as processes parallel to the main term:

$$\begin{array}{rcl} \llbracket M\langle x{:=}N\rangle\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, a & \triangleq & (\nu x)\,(\llbracket M\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, a \mid \llbracket x{:=}N\rrbracket^{\mathsf{L}}_{\mathsf{J}}) \;\triangleq\; (\nu x)\,(\llbracket M\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, a \mid !\overline{x}(w).\llbracket N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, w) \\ \llbracket M\langle \beta{:=}N{\cdot}\gamma\rangle\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, a & \triangleq & (\nu\beta)\,(\llbracket M\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, a \mid \llbracket \beta{:=}N{\cdot}\gamma\rrbracket^{\mathsf{L}}_{\mathsf{J}}) \;\triangleq\; (\nu\beta)\,(\llbracket M\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, a \mid !\beta(v,d).(\llbracket v{:=}N\rrbracket^{\mathsf{L}}_{\mathsf{J}} \mid !d{\to}\overline{\gamma})) \end{array}$$

This justifies the use of $\llbracket \boldsymbol{S}\rrbracket^{\mathsf{L}}_{\mathsf{J}}$ for the interpretation of a sequence of explicit substitutions and, if $\boldsymbol{S}$ defines $\vec{y}$ and $\vec{\alpha}$, then as a generalisation we can write $\llbracket M\boldsymbol{S}\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, a \equiv (\nu\vec{y}\vec{\alpha})\,(\llbracket M\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, a \mid \llbracket \boldsymbol{S}\rrbracket^{\mathsf{L}}_{\mathsf{J}})$.

We have, for example:

$$\begin{array}{rcl} \llbracket NM_1\boldsymbol{S}_1\cdots M_n\boldsymbol{S}_n\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, a & \triangleq & (\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})\,(\llbracket NM_1\boldsymbol{S}_1\cdots M_n\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, a \mid \llbracket \boldsymbol{S}_n\rrbracket^{\mathsf{L}}_{\mathsf{J}}) \\ & \triangleq & (\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})\,(\llbracket NM_1\boldsymbol{S}_1\cdots M_{n-1}\boldsymbol{S}_{n-1}M_n\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, a \mid \llbracket \boldsymbol{S}_n\rrbracket^{\mathsf{L}}_{\mathsf{J}}) \\ & \triangleq, \equiv & (\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})\cdots(\nu\overrightarrow{y_1}\overrightarrow{\alpha_1})\,(\nu\vec{c})\,(\llbracket N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, c_1 \mid \llbracket c_1{:=}(M_1){\cdot}c_2\rrbracket^{\mathsf{L}}_{\mathsf{J}} \mid \llbracket \boldsymbol{S}_1\rrbracket^{\mathsf{L}}_{\mathsf{J}} \mid \cdots \mid \llbracket c_n{:=}(M_n){\cdot}a\rrbracket^{\mathsf{L}}_{\mathsf{J}} \mid \llbracket \boldsymbol{S}_n\rrbracket^{\mathsf{L}}_{\mathsf{J}}) \\ & \equiv & (\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})\cdots(\nu\overrightarrow{y_1}\overrightarrow{\alpha_1})\,(\nu\vec{c})\,(\llbracket N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, c_1 \mid \llbracket c_1{:=}(M_1){\cdot}c_2\rrbracket^{\mathsf{L}}_{\mathsf{J}} \mid \cdots \mid \llbracket c_n{:=}(M_n){\cdot}a\rrbracket^{\mathsf{L}}_{\mathsf{J}} \mid \llbracket \boldsymbol{S}_1\rrbracket^{\mathsf{L}}_{\mathsf{J}} \mid \cdots \mid \llbracket \boldsymbol{S}_n\rrbracket^{\mathsf{L}}_{\mathsf{J}}) \\ & \equiv, \triangleq & \llbracket NM_1\cdots M_n\boldsymbol{S}_1\cdots\boldsymbol{S}_n\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, a \end{array}$$

Notice that, in the last step, the structural congruence forces the placement of the substitutions in the right order. This implies that, when dealing with interpreted application terms, we can safely assume all substitutions are placed on the outside. Under the right conditions (decided by the occurrences of variables and names) we can reverse these steps, so bring the substitutions inside, as illustrated in Ex. 6.4.

The operation of *renaming* we will use below is defined and justified via the following lemma, which states that we can safely rename the output of an interpreted $\lambda\mu$-term.

Lemma 6.7 (Renaming lemma) *Let e be a fresh name. Then*

i) If a is at most only used for output in $\llbracket M\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g$ and $a \neq g$, then $(\nu a)\,(!a{\to}\overline{e} \mid \llbracket M\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g) \;\approx\; \llbracket M\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g\,\{e/a\}$.

ii) If $a \notin M$, then $(\nu a)\,(!a{\to}\overline{e} \mid \llbracket M\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, a) \;\approx\; \llbracket M\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, e$.

Proof: By induction on the structure of $\lambda\mu\mathbf{x}$-terms.

$$\begin{array}{l} i) M = x\colon\ (\nu a)\,(!a{\to}\overline{e} \mid \llbracket x\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g) \;\triangleq\; (\nu a)\,(!a{\to}\overline{e} \mid x(u).!u{\to}\overline{g}) \;\equiv\; (\nu a)\,(!a{\to}\overline{e}) \mid x(u).!u{\to}\overline{g} \;\approx \\ \qquad x(u).!u{\to}\overline{g} \;\triangleq\; \llbracket x\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g \;=\; \llbracket x\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g\,\{e/a\} \\ M = \lambda x.N\colon\ (\nu a)\,(!a{\to}\overline{e} \mid \llbracket \lambda x.N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g) \;\triangleq\; (\nu a)\,(!a{\to}\overline{e} \mid (\nu xb)\,(\llbracket N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, b \mid \overline{g}\langle x,b\rangle)) \;\equiv\; (a \notin \overline{g}\langle x,b\rangle) \\ \qquad (\nu xb)\,((\nu a)\,(!a{\to}\overline{e} \mid \llbracket N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, b) \mid \overline{g}\langle x,b\rangle) \;\approx\; (IH) \quad (\nu xb)\,(\llbracket N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, b\,\{e/a\} \mid \overline{g}\langle x,b\rangle) \;\triangleq\; (a \notin \overline{g}\langle x,b\rangle) \\ \qquad \llbracket \lambda x.N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g\,\{e/a\} \\ M = PQ\colon\ (\nu a)\,(!a{\to}\overline{e} \mid \llbracket PQ\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g) \qquad \triangleq \\ \qquad (\nu a)\,(!a{\to}\overline{e} \mid (\nu c)\,(\llbracket P\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, c \mid c(v,d).(\llbracket v{:=}Q\rrbracket^{\mathsf{L}}_{\mathsf{J}} \mid !d{\to}\overline{g}) \mid \llbracket c{:=}Q{\cdot}g\rrbracket^{\mathsf{L}}_{\mathsf{J}})) \qquad \approx\ (2.5{:}1) \\ \qquad (\nu c)\,((\nu a)\,(!a{\to}\overline{e} \mid \llbracket P\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, c) \mid (\nu a)\,(!a{\to}\overline{e} \mid !c(v,d).(!\overline{v}(w).\llbracket Q\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, w \mid !d{\to}\overline{g}))) \;\approx\; (2.5{:}4) \\ \qquad (\nu c)\,((\nu a)\,(!a{\to}\overline{e} \mid \llbracket P\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, c) \mid !c(v,d).((\nu a)\,(!a{\to}\overline{e} \mid !\overline{v}(w).\llbracket Q\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, w \mid !d{\to}\overline{g}))) \;\approx\; (2.5{:}5) \\ \qquad (\nu c)\,((\nu a)\,(!a{\to}\overline{e} \mid \llbracket P\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, c) \mid !c(v,d).(!\overline{v}(w).(\nu a)\,(!a{\to}\overline{e} \mid \llbracket Q\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, w) \mid !d{\to}\overline{g})) \;\approx\; (IH) \\ \qquad (\nu c)\,(\llbracket P\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, c\,\{e/a\} \mid !c(v,d).(!\overline{v}(w).\llbracket Q\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, w\,\{e/a\} \mid !d{\to}\overline{g})) \qquad \triangleq\; \llbracket PQ\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g\,\{e/a\} \\ M = P\langle x{:=}Q\rangle\colon\ (\nu a)\,(!a{\to}\overline{e} \mid \llbracket P\langle x{:=}Q\rangle\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g) \qquad \triangleq \\ \qquad (\nu a)\,(!a{\to}\overline{e} \mid (\nu x)\,(\llbracket P\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g \mid !\overline{x}(w).\llbracket Q\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, w)) \qquad \approx\ (2.5{:}1,5) \\ \qquad (\nu x)\,((\nu a)\,(!a{\to}\overline{e} \mid \llbracket P\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g) \mid !\overline{x}(w).(\nu a)\,(!a{\to}\overline{e} \mid \llbracket Q\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, w)) \;\approx\; (IH) \\ \qquad (\nu x)\,(\llbracket P\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g\,\{e/a\} \mid !\overline{x}(w).\llbracket Q\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, w\,\{e/a\}) \qquad \triangleq\; \llbracket P\langle x{:=}Q\rangle\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g\,\{e/a\} \\ M = \mu\beta.[\beta]N\colon\ (\nu a)\,(!a{\to}\overline{e} \mid \llbracket \mu\beta.[\beta]N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g) \;\triangleq\; (\nu a)\,(!a{\to}\overline{e} \mid \llbracket N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, \beta\,\{g/\beta\}) \;\approx\; (IH) \\ \qquad \llbracket N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, \beta\,\{e/a\}\,\{g/\beta\} \qquad \triangleq\; \llbracket \mu\beta.[\beta]N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g\,\{e/a\} \\ M = \mu\beta.[\gamma]N,\ \beta \neq \gamma\colon\ (\nu a)\,(!a{\to}\overline{e} \mid \llbracket \mu\beta.[\gamma]N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g) \;\triangleq\; (\nu a)\,(!a{\to}\overline{e} \mid \llbracket N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, \gamma\,\{g/\beta\}) \;\approx (IH) \\ \qquad \llbracket N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, \gamma\,\{e/a\}\,\{g/\beta\} \;=(a{\neq}g)\; \llbracket N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, \gamma\,\{g/\beta\}\,\{e/a\} \;\triangleq\; \llbracket \mu\beta.[\gamma]N\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g\,\{e/a\} \\ \qquad\quad \text{Note that perhaps } a = \gamma. \\ M = P\langle \beta{:=}Q{\cdot}\gamma\rangle\colon\ (\nu a)\,(!a{\to}\overline{e} \mid \llbracket P\langle \beta{:=}Q{\cdot}\gamma\rangle\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g) \qquad \triangleq \\ \qquad (\nu a)\,(!a{\to}\overline{e} \mid (\nu\beta)\,(\llbracket P\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g \mid !\beta(v,d).(!\overline{v}(w).\llbracket Q\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, w \mid !d{\to}\gamma))) \;\approx\; (2.5{:}1,4) \\ \qquad (\nu\beta)\,((\nu a)\,(!a{\to}\overline{e} \mid \llbracket P\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g) \mid !\beta(v,d).(!\overline{v}(w).(\nu a)\,(!a{\to}\overline{e} \mid \llbracket Q\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, w) \mid !d{\to}\gamma)) \;\approx (IH) \\ \qquad (\nu\beta)\,(\llbracket P\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g\,\{e/a\} \mid !\beta(v,d).(!\overline{v}(w).\llbracket Q\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, w\,\{e/a\} \mid !d{\to}\gamma)) \;\triangleq\; \llbracket P\langle \beta{:=}Q{\cdot}\gamma\rangle\rrbracket^{\mathsf{L}}_{\mathsf{J}}\, g\,\{e/a\} \end{array}$$

$$
\begin{array}{ll}
ii) M = x\colon & (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket x \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a) \;\triangleq\; (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid x(u).!\,u{\rightarrow}\overline{a}) \;\approx (2.5{:}3) \\
& x(u).((\nu a)\,(!\,a{\rightarrow}\overline{e} \mid !\,u{\rightarrow}\overline{a})) \;\approx\; x(u).!\,u{\rightarrow}\overline{e} \;\triangleq\; \llbracket x \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, e \\
M = \lambda x.N\colon & (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket \lambda x.N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a) \;\triangleq\; (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid (\nu x b)\,(\llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, b \mid \overline{a}\langle x,b\rangle)) \;\equiv \\
& (\nu a x b)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, b \mid \overline{a}\langle x,b\rangle) \;\equiv\; (a \notin \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, b) \\
& (\nu x b)\,(\llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, b \mid (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \overline{a}\langle x,b\rangle)) \;\approx\; (2.6{:}11) \\
& (\nu x b)\,(\llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, b \mid \overline{e}\langle x,b\rangle) \;\triangleq\; \llbracket \lambda x.N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, e \\
M = PQ\colon & (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket PQ \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a) \;\triangleq \\
& (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid (\nu c)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid c(v,d).(\llbracket v{:=}Q \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d{\rightarrow}\overline{a}) \mid \llbracket c{:=}Q{\cdot}a \rrbracket^{\mathrm{L}}_{\mathrm{J}})) \;\approx\; (2.5{:}1,4,5) \\
& (\nu c)\,((\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c) \mid !\,c(v,d).(!\,\overline{v}(w).(\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket Q \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w) \mid (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid !\,d{\rightarrow}\overline{a}))) \\
& \qquad\qquad \approx\; (a \notin \llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c, \llbracket Q \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w) \\
& (\nu c)\,((\nu a)\,(!\,a{\rightarrow}\overline{e}) \mid \llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid !\,c(v,d).(!\,\overline{v}(w).(\nu a)\,(!\,a{\rightarrow}\overline{e}) \mid \llbracket Q \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w \mid (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid !\,d{\rightarrow}\overline{a}))) \approx \\
& (\nu c)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid c(v,d).(\llbracket v{:=}Q \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d{\rightarrow}\overline{e}) \mid \llbracket c{:=}Q{\cdot}e \rrbracket^{\mathrm{L}}_{\mathrm{J}}) \;\triangleq \\
& \llbracket PQ \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, e \\
M = P\langle x{:=}Q\rangle\colon & (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket P\langle x{:=}Q\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a) \;\triangleq \\
& (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid (\nu x)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a \mid !\,\overline{x}(w).\llbracket Q \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w)) \;\equiv\; (x \neq a,\ a \notin !\,\overline{x}(w).\llbracket Q \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w) \\
& (\nu x)\,((\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a) \mid !\,\overline{x}(w).\llbracket Q \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w) \;\approx\; (IH) \\
& (\nu x)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, e \mid !\,\overline{x}(w).\llbracket Q \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w) \;\triangleq\; \llbracket P\langle x{:=}Q\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, e \\
M = \mu\beta.[\beta]N\colon & (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket \mu\beta.[\beta]N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a) \;\triangleq\; (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, \beta\,\{a/\beta\}) \;= \\
& (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a) \;\approx (IH)\; \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, e \;=\; \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, \beta\,\{e/\beta\} \;\triangleq\; \llbracket \mu\beta.[\beta]N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, e \\
M = \mu\beta.[\gamma]N,\ \beta \neq \gamma\colon & (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket \mu\beta.[\gamma]N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a) \;\triangleq\; (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, \gamma\,\{a/\beta\}) \;=_{\alpha} \\
& (\nu\beta)\,(!\,\beta{\rightarrow}\overline{e} \mid \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, \gamma) \;\approx (part\ (i))\; \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, \gamma\,\{e/\beta\} \;\triangleq\; \llbracket \mu\beta.[\gamma]N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, e \\
M = P\langle \beta{:=}Q{\cdot}\gamma\rangle\colon & (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket P\langle \beta{:=}Q{\cdot}\gamma\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a) \;\triangleq \\
& (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid (\nu\beta)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a \mid !\,\beta(v,d).(!\,\overline{v}(w).\llbracket Q \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w \mid !\,d{\rightarrow}\gamma))) \;\approx\; (2.5{:}1) \\
& (\nu\beta)\,((\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a) \mid (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid !\,\beta(v,d).(\llbracket v{:=}Q \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d{\rightarrow}\overline{\gamma}))) \;\equiv \\
& \qquad\qquad (a \notin !\,\beta(v,d).(\llbracket v{:=}Q \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d{\rightarrow}\overline{\gamma})) \\
& (\nu\beta)\,((\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a) \mid (\nu a)\,(!\,a{\rightarrow}\overline{e}) \mid !\,\beta(v,d).(\llbracket v{:=}Q \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d{\rightarrow}\overline{\gamma})) \;\approx\; (IH) \\
& (\nu\beta)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, e \mid !\,\beta(v,d).(\llbracket v{:=}Q \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d{\rightarrow}\overline{\gamma})) \;\triangleq\; \llbracket P\langle \beta{:=}Q{\cdot}\gamma\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, e
\end{array}
$$

□

For reasons of clarity, we use some auxiliary notions of equivalence, that are used in Thm 7.1.

Since we will be dealing with explicit substitution rather than implicit substitution as Milner did, some of the results we will show below will use a weaker variant of that relation, being '$\approx_{\mathrm{G}}$' (garbage collection), '$\approx_{\mathrm{R}}$' (renaming), and '$\approx_{\mathrm{D}}$' (duplication).

Definition 6.8 *i*) We call a process that is weakly bisimilar to *0 garbage*. We define a *garbage collection* bisimilarity by: $P \approx_{\mathrm{G}} Q$ if and only if there exists R such that $P \equiv Q \mid R$ and $R \approx 0$.

ii) We define '$\approx_{\mathrm{R}}$' (*renaming*) as the smallest equivalence such that:

a) for all M: $(\nu a)\,(\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a \mid !\,a{\rightarrow}\overline{e}) \approx_{\mathrm{R}} \llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, e$.

b) for all M: if $a \neq b$, then $(\nu a)\,(\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, b \mid !\,a{\rightarrow}\overline{e}) \approx_{\mathrm{R}} \llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, b\,\{e/a\}$.

c) if $P \approx_{\mathrm{R}} Q$, then $(\nu\vec{b})\,(P \mid R) \approx_{\mathrm{R}} (\nu\vec{b})\,(Q \mid R)$.

iii) We define '$\approx_{\mathrm{D}}$' (*distribution*) as the smallest equivalence such that:

a) for all M, N:

$$
\begin{array}{l}
(\nu\alpha)\,(\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, \alpha \mid !\,\alpha(v,d).(\llbracket v{:=}N \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d{\rightarrow}\overline{\gamma})) \;\approx_{\mathrm{D}} \\
\quad (\nu\alpha)\,((\nu c)\,(\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid !\,c(v,d).(\llbracket v{:=}N \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d{\rightarrow}\overline{\gamma})) \mid !\,\alpha(v,d).(\llbracket v{:=}N \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d{\rightarrow}\overline{\gamma}))\ (c\ fresh)
\end{array}
$$

Notice that we 'split' the substitution $!\,\alpha(v,d).(\llbracket v{:=}N \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d{\rightarrow}\overline{\gamma})$ in two parts: one dealing with the outermost name α (that for the whole term) which gets renamed to c, and one dealing with the remaining occurrences of α in $\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c$.

b) if $P \approx_{\mathrm{D}} Q$, then $(\nu\vec{b})\,(P \mid R) \approx_{\mathrm{D}} (\nu\vec{b})\,(Q \mid R)$.

iv) We define '$\approx_{\mathrm{RGD}}$' $\triangleq$ '$\approx_{\mathrm{R}}, \approx_{\mathrm{G}}, \approx_{\mathrm{D}}$' (so applied left-to-right); '$\approx_{\mathrm{RG}}$' $\triangleq$ '$\approx_{\mathrm{R}}, \approx_{\mathrm{G}}$'; and '$\approx_{\mathrm{GD}}$' $\triangleq$

'$\approx_{\mathrm{G}}, \approx_{\mathrm{D}}$', where each '$\approx_{[\cdot]}$' component can be omitted.

So '$\approx_{\mathrm{R}}$' is used when we want to emphasise that two processes are equivalent just using renaming. Notice that renaming and distribution are not allowed under guard. Moreover, '$\approx_{\mathrm{G}}$' $\subset$ '$\approx$', '$\approx_{\mathrm{R}}$' $\subset$ '$\approx$' by Proposition 6.7, and that '$\approx_{\mathrm{D}}$' $\subset$ '$\approx$' by Thm 2.5:2.

Using the Renaming Lemma, we can show the following:

Example 6.9 The interpretation of the β-redex $(\lambda x.M)\,N$ reduces as follows:

$$\begin{array}{rcl}
\llbracket(\lambda x.M)\,N\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a & \triangleq & (\nu c)\,((\nu xb)\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,b \mid \overline{c}\langle x,b\rangle) \mid !c(v,d).(\llbracket v{:=}N\rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !d{\to}\overline{a})) \\
 & \to_{\pi}(c),\equiv & (\nu bx)\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,b \mid !b{\to}\overline{a} \mid \llbracket x{:=}N\rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid (\nu c)\,(!c(v,d).(\llbracket v{:=}N\rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !d{\to}\overline{a})) \\
 & \approx_{\mathrm{R}}(6.7) & (\nu x)\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a \mid \llbracket x{:=}N\rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid (\nu c)\,(!c(v,d).(\llbracket v{:=}N\rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !d{\to}\overline{a})) \\
 & \approx_{\mathrm{G}}(*) & (\nu x)\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a \mid \llbracket x{:=}N\rrbracket^{\mathrm{L}}_{\mathrm{J}}) \\
 & \triangleq & \llbracket M\langle x{:=}N\rangle\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a
\end{array}$$

This shows that each β-reduction step is implemented in π by at least one synchronisation. Notice that, in step $(*)$, the process $(\nu c)\,(!c(v,d).(\llbracket v{:=}N\rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !d{\to}\overline{a}))$ is weakly bisimilar to $\mathit{0}$. Moreover, the synchronisation over c is over a hidden channel, so by Proposition 2.6:11 we can conclude $\llbracket(\lambda x.M)N\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a \approx \llbracket M\langle x{:=}N\rangle\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a$.

Since $\llbracket M\langle x{:=}N\rangle\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a$ places $\llbracket M\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a$ and $\llbracket x := N\rrbracket^{\mathrm{L}}_{\mathrm{J}}$ in parallel, using Lemma 2.5 we can even show that the explicit variant of the Substitution Lemma is preserved:

Lemma 6.10 (Substitution Lemma) $\llbracket M\langle y{:=}N\rangle\,\langle x{:=}L\rangle\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a \;\approx\; \llbracket M\langle x{:=}L\rangle\,\langle y{:=}N\langle x{:=}L\rangle\rangle\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a$.

Proof: We can assume $x \neq y$, and $x \notin N$, $y \notin L$.

$$\begin{array}{rcl}
\llbracket M\langle y{:=}N\rangle\,\langle x{:=}L\rangle\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a & \triangleq & \\
(\nu x)\,((\nu y)\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a \mid !\overline{y}(w).\llbracket N\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,w) \mid !\overline{x}(w).\llbracket L\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,w) & \approx & (2.5{:}6) \\
(\nu y)\,((\nu x)\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a \mid !\overline{x}(w).\llbracket L\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,w) \mid (\nu x)\,(!\overline{y}(w).\llbracket N\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,w \mid !\overline{x}(w).\llbracket L\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,w)) & \approx & (2.5{:}9) \\
(\nu y)\,((\nu x)\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a \mid !\overline{x}(w).\llbracket L\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,w) \mid !\overline{y}(w).(\nu x)\,(\llbracket N\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,y \mid !\overline{x}(w).\llbracket L\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,w)) & \triangleq & \\
(\nu y)\,(\llbracket M\langle x{:=}L\rangle\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a \mid !\overline{y}(w).\llbracket N\langle x{:=}L\rangle\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,w) & \triangleq & \\
\llbracket M\langle x{:=}L\rangle\,\langle y{:=}N\langle x{:=}L\rangle\rangle\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a & &
\end{array}$$

□

7 Soundness, completeness, and termination

As in [43, 52], we can now show a reduction-preservation result for our encoding with respect to explicit head-reduction for $\lambda\mu\mathbf{x}$, by showing that $\llbracket\cdot\rrbracket^{\mathrm{L}}_{\mathrm{J}}\cdot$ preserves '$\to_{\mathrm{XH}}$' up to weak bisimilarity (mainly through garbage collection and/or renaming). Notice that we prove the result for $\lambda\mu\mathbf{x}$ terms, do not require the terms to be closed, and that the result is shown for single step reduction.

Theorem 7.1 (Soundness) *If $P \to_{\mathrm{XH}} Q$, then there exist R such that $\llbracket P\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a \to^{*}_{\pi} R$ and $R \approx_{\mathrm{RGD}} \llbracket Q\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a$ (i.e.$\llbracket P\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a \to^{*}_{\pi}, \approx_{\mathrm{RGD}} \llbracket Q\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a$).*

Proof: By induction on the definition of explicit head-reduction.

Main reduction rules : (β): Then $P = (\lambda x.M)\,N$ and $Q = M\langle x{:=}N\rangle$; by Example 6.9.

(μ_p): Then $P = (\mu\alpha.[\alpha]M)N$, $Q = \mu\gamma.[\gamma]M\langle\alpha{:=}N{\cdot}\gamma\rangle N$, with γ fresh, and

$$\begin{array}{rcl}
\llbracket(\mu\alpha.[\alpha]M)\,N\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a \;\triangleq\; (\nu c)\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,\alpha\,\{c/\alpha\} \mid !c(v,d).(\llbracket v{:=}N\rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !d{\to}\overline{a})) & \approx_{\mathrm{D}} & \\
(\nu c)\,((\nu\alpha)\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,c \mid \llbracket\alpha{:=}N{\cdot}a\rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid \llbracket c{:=}N{\cdot}a\rrbracket^{\mathrm{L}}_{\mathrm{J}}) & = & \\
(\nu c)\,((\nu\alpha)\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,c \mid \llbracket\alpha{:=}N{\cdot}\gamma\rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid \llbracket c{:=}N{\cdot}\gamma\rrbracket^{\mathrm{L}}_{\mathrm{J}})\,\{a/\gamma\} & \triangleq & \\
(\nu c)\,(\llbracket M\langle\alpha{:=}N{\cdot}\gamma\rangle\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,c \mid \llbracket c{:=}N{\cdot}\gamma\rrbracket^{\mathrm{L}}_{\mathrm{J}})\,\{a/\gamma\} & \triangleq & \\
\llbracket M\langle\alpha{:=}N{\cdot}\gamma\rangle N\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,\gamma\,\{a/\gamma\} \;\triangleq\; \llbracket\mu\gamma.[\gamma]M\langle\alpha{:=}N{\cdot}\gamma\rangle N\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a & &
\end{array}$$

(μ_r): Then $P = (\mu\alpha.[\beta]M)N$, $Q = \mu\alpha.[\beta]M\langle\alpha{:=}N{\cdot}\gamma\rangle N$, with $\alpha \neq \beta$, and $\llbracket(\mu\alpha.[\beta]M)\,N\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a \;\triangleq$
$(\nu c)\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,\beta\,\{c/\alpha\} \mid \llbracket c{:=}N{\cdot}a\rrbracket^{\mathrm{L}}_{\mathrm{J}}) \;=_{\alpha}\; (c\ \mathit{fresh})\quad (\nu\alpha)\,(\llbracket M\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,\beta \mid \llbracket\alpha{:=}N{\cdot}\gamma\rrbracket^{\mathrm{L}}_{\mathrm{J}})\,\{a/\gamma\} \;\triangleq$
$\llbracket M\langle\alpha{:=}N{\cdot}\gamma\rangle\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,\beta\,\{a/\gamma\} \;\triangleq\; \llbracket\mu\gamma.[\beta]M\langle\alpha{:=}N{\cdot}\gamma\rangle\rrbracket^{\mathrm{L}}_{\mathrm{J}}\,a$

(R): Then $P = \mu\alpha.[\beta]\mu\gamma.[\delta]M$ and $Q = \mu\alpha.([\delta]M)\{\beta/\gamma\}$. We distinguish:

$\delta = \gamma$: $\llbracket \mu\alpha.[\beta]\mu\gamma.[\gamma]M \rrbracket^{\mathrm{L}} a \;\triangleq\; \llbracket \mu\gamma.[\gamma]M \rrbracket^{\mathrm{L}} \beta\{a/\alpha\} \;\triangleq\; \llbracket M \rrbracket^{\mathrm{L}} \gamma\{\beta/\gamma\}\{a/\alpha\} \;=$
$\llbracket M\{\beta/\gamma\} \rrbracket^{\mathrm{L}} \beta\{a/\alpha\} \;\triangleq\; \llbracket \mu\alpha.[\beta]M\{\beta/\gamma\} \rrbracket^{\mathrm{L}} a$

$\delta \neq \gamma$: $\llbracket \mu\alpha.[\beta]\mu\gamma.[\delta]M \rrbracket^{\mathrm{L}} a \;\triangleq\; \llbracket \mu\gamma.[\delta]M \rrbracket^{\mathrm{L}} \beta\{a/\alpha\} \;\triangleq\; \llbracket M \rrbracket^{\mathrm{L}} \delta\{\beta/\gamma\}\{a/\alpha\} \;=$
$\llbracket M\{\beta/\gamma\} \rrbracket^{\mathrm{L}} \delta\{a/\alpha\} \;\triangleq\; \llbracket \mu\alpha.[\delta]M\{\beta/\gamma\} \rrbracket^{\mathrm{L}} a$

(C): Then $P = \mu\alpha.[\alpha]M$, $Q = M$, with $\alpha \notin fn(M)$, and $\llbracket \mu\alpha.[\alpha]M \rrbracket^{\mathrm{L}} a \;\triangleq\; \llbracket M \rrbracket^{\mathrm{L}} \alpha\{a/\alpha\} \;= (\alpha \notin fn(M))$ $\llbracket M \rrbracket^{\mathrm{L}} a$.

Substitution rules : (hv): Then $P = x\boldsymbol{S}_0 M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n \langle x{:=}N\rangle$ and $Q = N\boldsymbol{S}_0 M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n \langle x{:=}N\rangle$. Take $\boldsymbol{S} = \boldsymbol{S}_0 \cdots \boldsymbol{S}_n$.

$\llbracket x\boldsymbol{S}_0 M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n \langle x{:=}N\rangle \rrbracket^{\mathrm{L}} a \;\equiv (6.6)\; (\nu x)(\llbracket xM_1 \cdots M_n \boldsymbol{S}^{\mathrm{L}} a \mid \llbracket x{:=}N \rrbracket^{\mathrm{L}}) \;\triangleq$
$(\nu x)((\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})((\nu c)(\llbracket xM_1 \cdots M_{n-1} \rrbracket^{\mathrm{L}} c \mid \llbracket c{:=}(M_n)\cdot a \rrbracket^{\mathrm{L}}) \mid \llbracket \boldsymbol{S}^{\mathrm{L}} \rrbracket) \mid \llbracket x{:=}N \rrbracket^{\mathrm{L}}) \;\triangleq\; (c_{n+1} = a)$
$(\nu x)((\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})(\llbracket x \rrbracket^{\mathrm{L}} c_1 \mid \overrightarrow{\llbracket c_i{:=}M_i\cdot c_{i+1} \rrbracket^{\mathrm{L}}} \mid \llbracket \boldsymbol{S}^{\mathrm{L}} \rrbracket) \mid \llbracket x{:=}N \rrbracket^{\mathrm{L}}) \;\triangleq, \equiv$
$(\nu x)((\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})(x(u).!u{\to}\overline{c_1} \mid \overline{x}(w).\llbracket N \rrbracket^{\mathrm{L}} w \mid \overrightarrow{\llbracket c_i{:=}M_i\cdot c_{i+1} \rrbracket^{\mathrm{L}}} \mid \llbracket \boldsymbol{S}^{\mathrm{L}} \rrbracket) \mid \llbracket x{:=}N \rrbracket^{\mathrm{L}}) \;\to_\pi (x)$
$(\nu x)((\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})((\nu w)(\llbracket N \rrbracket^{\mathrm{L}} w \mid !w{\to}\overline{c_1}) \mid \overrightarrow{\llbracket c_i{:=}M_i\cdot c_{i+1} \rrbracket^{\mathrm{L}}} \mid \llbracket \boldsymbol{S}^{\mathrm{L}} \rrbracket) \mid \llbracket x{:=}N \rrbracket^{\mathrm{L}}) \;\approx_{\mathrm{R}}$
$(\nu x)((\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})(\llbracket N \rrbracket^{\mathrm{L}} c_1 \mid \overrightarrow{\llbracket c_i{:=}M_i\cdot c_{i+1} \rrbracket^{\mathrm{L}}} \mid \llbracket \boldsymbol{S}^{\mathrm{L}} \rrbracket) \mid \llbracket x{:=}N \rrbracket^{\mathrm{L}}) \;\triangleq$
$\llbracket NM_1 \cdots M_n \boldsymbol{S}\langle x{:=}N\rangle \rrbracket^{\mathrm{L}} a \;\equiv\; \llbracket N\boldsymbol{S}_0 M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n \langle x{:=}N\rangle \rrbracket^{\mathrm{L}} a$

Notice that the substitutions can be moved back in position in the last step since the binding of variables and names has not changed; the only change is that N takes the position of x.

$(\lambda\boldsymbol{S})$: Then $P = (\lambda y.M)\boldsymbol{S}$, $Q = \lambda x.M\boldsymbol{S}$, and $\llbracket (\lambda x.M)\boldsymbol{S}^{\mathrm{L}} a \;\triangleq\; (\nu\vec{y}\vec{\alpha})((\nu xb)(\llbracket M \rrbracket^{\mathrm{L}} b \mid \overline{a}\langle x,b\rangle) \mid \llbracket \boldsymbol{S}^{\mathrm{L}} \rrbracket)$
$\equiv (a \notin \vec{y}\vec{\alpha})\;\; (\nu xb)((\nu\vec{y}\vec{\alpha})(\llbracket M \rrbracket^{\mathrm{L}} b \mid \llbracket \boldsymbol{S}^{\mathrm{L}} \rrbracket) \mid \overline{a}\langle y,b\rangle) \;\triangleq\; \llbracket \lambda x.(M\boldsymbol{S}) \rrbracket^{\mathrm{L}} a$

(hn): Then $P = (\mu\delta.[\beta]M)\boldsymbol{S}\langle\beta{:=}N\cdot\gamma\rangle$ and $Q = \mu\delta.[\gamma](M\boldsymbol{S}\langle\beta{:=}N\cdot\gamma\rangle)N$. Then

$\llbracket (\mu\delta.[\beta]M)\boldsymbol{S}\langle\beta{:=}N\cdot\gamma\rangle \rrbracket^{\mathrm{L}} a \;\triangleq\; (\nu\beta)(\llbracket (\mu\delta.[\beta]M)\boldsymbol{S}^{\mathrm{L}} a \mid !\beta(v,d).(\llbracket v{:=}N \rrbracket^{\mathrm{L}} \mid !d{\to}\overline{\gamma})) \;\triangleq$
$(\nu\beta)((\nu\vec{y}\vec{\alpha})(\llbracket \mu\delta.[\beta]M \rrbracket^{\mathrm{L}} a \mid \llbracket \boldsymbol{S}^{\mathrm{L}} \rrbracket) \mid !\beta(v,d).(\llbracket v{:=}N \rrbracket^{\mathrm{L}} \mid !d{\to}\overline{\gamma})) \;\triangleq$
$(\nu\beta)((\nu\vec{y}\vec{\alpha})(\llbracket M \rrbracket^{\mathrm{L}} \beta\{a/\delta\} \mid \llbracket \boldsymbol{S}^{\mathrm{L}} \rrbracket) \mid !\beta(v,d).(\llbracket v{:=}N \rrbracket^{\mathrm{L}} \mid !d{\to}\overline{\gamma})) \;\approx_{\mathrm{D}}$
$\llbracket dcZM\boldsymbol{S}\beta := N.\gamma N \rrbracket^{\mathrm{L}} \gamma\{a/\delta\} \;\triangleq$
$(\nu c)((\nu\beta)(\llbracket M\boldsymbol{S}^{\mathrm{L}} c \mid !\beta(v,d).(\llbracket v{:=}N \rrbracket^{\mathrm{L}} \mid !d{\to}\overline{\gamma})) \mid !c(v,d).(\llbracket v{:=}N \rrbracket^{\mathrm{L}} \mid !d{\to}\overline{\gamma}))\{a/\delta\} \;\triangleq$
$(\nu c)(\llbracket M\boldsymbol{S}\langle\beta{:=}N\cdot\gamma\rangle \rrbracket^{\mathrm{L}} c \mid !c(v,d).(\llbracket v{:=}N \rrbracket^{\mathrm{L}} \mid !d{\to}\overline{\gamma}))\{a/\delta\} \;\triangleq$
$\llbracket (M\boldsymbol{S}\langle\beta{:=}N\cdot\gamma\rangle)N \rrbracket^{\mathrm{L}} \gamma\{a/\delta\} \;\triangleq\; \llbracket \mu\delta.[\gamma](M\boldsymbol{S}\langle\beta{:=}N\cdot\gamma\rangle)N \rrbracket^{\mathrm{L}} a$

$(n\boldsymbol{S})$: Then $P = (\mu\delta.[\beta]M)\boldsymbol{S}$ and $Q = \mu\delta.[\beta]M\boldsymbol{S}$, provided $\beta \notin \boldsymbol{S}$, and $\llbracket (\mu\delta.[\beta]M)\boldsymbol{S}^{\mathrm{L}} a \;\triangleq$ $(\nu\vec{y}\vec{\alpha})(\llbracket M \rrbracket^{\mathrm{L}} \beta\{a/\delta\} \mid \llbracket \boldsymbol{S}^{\mathrm{L}} \rrbracket) \;= (\delta \notin \boldsymbol{S})\;\; (\nu\vec{y}\vec{\alpha})(\llbracket M \rrbracket^{\mathrm{L}} \beta \mid \llbracket \boldsymbol{S}^{\mathrm{L}} \rrbracket)\{a/\delta\} \;\triangleq\; \llbracket M\boldsymbol{S}^{\mathrm{L}} \beta\{a/\delta\} \;\triangleq$ $\llbracket \mu\delta.[\beta]M\boldsymbol{S}^{\mathrm{L}} a$

(gc): Then $P = M\boldsymbol{S}$ and $Q = M\boldsymbol{S}\backslash c$, provided $c \in \boldsymbol{S}$, $c \notin M$, and $\llbracket M\boldsymbol{S}^{\mathrm{L}} a \;\triangleq\; (\nu\vec{y}\vec{\alpha})(\llbracket M \rrbracket^{\mathrm{L}} a \mid \llbracket \boldsymbol{S}^{\mathrm{L}} \rrbracket) \;\triangleq$
$(\nu\vec{y}\vec{\alpha})(\llbracket M \rrbracket^{\mathrm{L}} a \mid \llbracket \boldsymbol{S}_c \rrbracket^{\mathrm{L}} \mid \llbracket \boldsymbol{S}\backslash c \rrbracket^{\mathrm{L}}) \;\equiv\; (\nu\vec{y}\vec{\alpha})(\llbracket M \rrbracket^{\mathrm{L}} a \mid (\nu c)(\llbracket \boldsymbol{S}_c \rrbracket^{\mathrm{L}}) \mid \llbracket \boldsymbol{S}\backslash c \rrbracket^{\mathrm{L}}) \;\approx_{\mathrm{G}}$
$(\nu\vec{y}\vec{\alpha})(\llbracket M \rrbracket^{\mathrm{L}} a \mid \llbracket \boldsymbol{S}\backslash c \rrbracket^{\mathrm{L}}) \;\triangleq\; \llbracket M\boldsymbol{S}\backslash c \rrbracket^{\mathrm{L}} a$

Remember that $(\nu c)(\llbracket \boldsymbol{S}_c \rrbracket^{\mathrm{L}}) \approx 0$.

Contextual rules : $M \to N \Rightarrow ML \to NL$: $\llbracket ML \rrbracket^{\mathrm{L}} a \;\triangleq\; (\nu c)(\llbracket M \rrbracket^{\mathrm{L}} c \mid \llbracket c{:=}L\cdot a \rrbracket^{\mathrm{L}}) \;\to^*_\pi, \approx_{\mathrm{RGD}} (IH)$ $(\nu c)(\llbracket N \rrbracket^{\mathrm{L}} c \mid \llbracket c{:=}L\cdot a \rrbracket^{\mathrm{L}}) \;\triangleq\; \llbracket NL \rrbracket^{\mathrm{L}} a$

$M \to N \Rightarrow \lambda x.M \to \lambda x.N$: $\llbracket \lambda x.M \rrbracket^{\mathrm{L}} a \;\triangleq\; (\nu xb)(\llbracket M \rrbracket^{\mathrm{L}} b \mid \overline{a}\langle x,b\rangle) \;\to^*_\pi, \approx_{\mathrm{RGD}} (IH)$ $(\nu xb)(\llbracket N \rrbracket^{\mathrm{L}} b \mid \overline{a}\langle x,b\rangle) \;\triangleq\; \llbracket \lambda x.N \rrbracket^{\mathrm{L}} a$

$M \to N \Rightarrow \mu\alpha.[\beta]M \to \mu\alpha.[\beta]N$: $\llbracket \mu\alpha.[\beta]M \rrbracket^{\mathrm{L}} a \;\triangleq\; \llbracket M \rrbracket^{\mathrm{L}} \beta\{a/\alpha\} \;\to^*_\pi, \approx_{\mathrm{RGD}} (IH)\; \llbracket N \rrbracket^{\mathrm{L}} \beta\{a/\alpha\} \;\triangleq$ $\llbracket \mu\alpha.[\beta]N \rrbracket^{\mathrm{L}} a$

$M \to N \Rightarrow M\langle x{:=}L\rangle \to N\langle x{:=}L\rangle$: $\llbracket M\langle x{:=}L\rangle \rrbracket^{\mathrm{L}} a \;\triangleq\; (\nu x)(\llbracket M \rrbracket^{\mathrm{L}} a \mid \llbracket x{:=}L \rrbracket^{\mathrm{L}}) \;\to^*_\pi, \approx_{\mathrm{RGD}} (IH)$ $(\nu x)(\llbracket N \rrbracket^{\mathrm{L}} a \mid \llbracket x{:=}L \rrbracket^{\mathrm{L}}) \;\triangleq\; \llbracket N\langle x{:=}L\rangle \rrbracket^{\mathrm{L}} a$

$M \to N \Rightarrow M\langle\alpha{:=}L\cdot\gamma\rangle \to N\langle\alpha{:=}L\cdot\gamma\rangle$: $\llbracket M\langle\alpha{:=}L\cdot\gamma\rangle \rrbracket^{\mathrm{L}} a \;\triangleq\; (\nu\alpha)(\llbracket M \rrbracket^{\mathrm{L}} a \mid \llbracket \alpha{:=}L\cdot\gamma \rrbracket^{\mathrm{L}})$ $\to^*_\pi, \approx_{\mathrm{RGD}} (IH)\; (\nu\alpha)(\llbracket N \rrbracket^{\mathrm{L}} a \mid \llbracket \alpha{:=}L\cdot\gamma \rrbracket^{\mathrm{L}}) \;\triangleq\; \llbracket N\langle\alpha{:=}L\cdot\gamma\rangle \rrbracket^{\mathrm{L}} a$

□

Notice that, in the inductive cases, we do not have to deal with processes under guard, so do not need the full power of '$\approx$', as is needed for example for Thm 7.6 and 7.7, or as Milner and

Sangiorgi needed when modelling implicit substitution (see Sect. 3).

Notice that we need Lemma 2.5 only for renaming and to model the distribution of the contextual substitution $\langle \beta := N \cdot \gamma \rangle$ in the rules (μ_p) and (hn).

Remark that in the proof of Thm 7.1, the reduction rules (R) and $(\lambda \boldsymbol{S})$ are modelled using '$\equiv$', and that the rules (μ_r), (C) and $(n\boldsymbol{S})$ are dealt with by the interpretation directly. That leaves only five rules where '$\approx$' plays a role:

(β): through '$\to_\pi$', '$\approx_{\mathrm{R}}$' (which might include a '$\to_\pi$' step) and '$\approx_{\mathrm{G}}$';

(μ_p): through '$\approx_{\mathrm{D}}$';

(hv): through '$\to_\pi$' and '$\approx_{\mathrm{R}}$';

(hn): through '$\approx_{\mathrm{D}}$'; and

(gc): through '$\approx_{\mathrm{G}}$'.

Moreover, '$\approx_{\mathrm{R}}$' (as far as not a synchronisation itself propagating through the forwarders), '$\approx_{\mathrm{G}}$', and '$\approx_{\mathrm{D}}$' can be postponed until last, and do not interfere with the synchronisations.

We can thereby easily show:

Theorem 7.2 (Operational Soundness for '$\to_{\mathrm{XH}}$') *i) If $M \to^*_{\mathrm{XH}} N$, then $\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}} a \to^*_{\pi,\approx_{\mathrm{RGD}}} \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}} a$.*
ii) If $M \Uparrow_{\mathrm{XH}}$ then $\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}} a \Uparrow_\pi$.

Proof: The first is shown by induction on the length of the reduction sequence, using Thm 7.1. The second follows from the fact β-reduction is implemented in π by at least one synchronisation, as shown in Example 6.9, and that μ-reduction terminates [50], as does explicit substitution, so non-termination is caused only by β-reduction. □

By Prop. 2.6, all proper synchronisations in this proof are in '$\approx$'; this implies that, as far as the proof is concerned, we could have used '$\approx$' instead of '$\to^*_{\pi,\approx_{\mathrm{RGD}}}$'.

This implies that the following is immediate:

Theorem 7.3 *If $M \to^*_{\mathrm{XH}} N$, then $\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}} a \approx \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}} a$.*

Remark that we could not have represented the extensional rules: note that

$$\llbracket \lambda x.yx \rrbracket^{\mathrm{L}}_{\mathrm{J}} a \;\triangleq\; (\nu xb)\,((\nu c)\,(\llbracket y \rrbracket^{\mathrm{L}}_{\mathrm{J}} c \mid !c(v,d).(\llbracket v := x \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !d \to \overline{b})) \mid \overline{a}\langle x,b\rangle)$$

is not weakly bisimilar to $\llbracket y \rrbracket^{\mathrm{L}}_{\mathrm{J}} a$, and neither is

$$\llbracket \mu\alpha.[\beta]y \rrbracket^{\mathrm{L}}_{\mathrm{J}} a \;\triangleq\; \llbracket y \rrbracket^{\mathrm{L}}_{\mathrm{J}} \beta\,\{a/\alpha\} \;=\; \llbracket y \rrbracket^{\mathrm{L}}_{\mathrm{J}} \beta$$

weakly bisimilar to:

$$\begin{array}{lllll} \llbracket \lambda x.\mu\gamma.[\beta]y\{x\cdot\gamma/\alpha\} \rrbracket^{\mathrm{L}}_{\mathrm{J}} a & = & \llbracket \lambda x.\mu\gamma.[\beta]y \rrbracket^{\mathrm{L}}_{\mathrm{J}} a & \triangleq & (\nu xb)\,(\llbracket \mu\gamma.[\beta]y \rrbracket^{\mathrm{L}}_{\mathrm{J}} b \mid \overline{a}\langle x,b\rangle) \\ & \triangleq & (\nu xb)\,(\llbracket y \rrbracket^{\mathrm{L}}_{\mathrm{J}} \beta\,\{b/\gamma\} \mid \overline{a}\langle x,b\rangle) & \triangleq & (\nu xb)\,(\llbracket y \rrbracket^{\mathrm{L}}_{\mathrm{J}} \beta \mid \overline{a}\langle x,b\rangle) \end{array}$$

Remember that we have

$$(\lambda x.(\lambda y.M)\,N)\,L \;\to\; \begin{cases} (\lambda x.(M\langle y := N\rangle))\,L \\ ((\lambda y.M)\,N)\,\langle x := L\rangle \end{cases}$$

and in the process

$$\begin{array}{lll} \llbracket (\lambda x.(\lambda y.M)\,N)\,L \rrbracket^{\mathrm{L}}_{\mathrm{J}} a & \triangleq & (\nu e)\,((\nu xb)\,((\nu c)\,((\nu yb)\,(\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}} b \mid \overline{c}\langle y,b\rangle) \mid \\ & & \qquad \llbracket c := N\cdot b \rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid \overline{e}\langle x,b\rangle) \mid !e(v,d).(\llbracket v := L \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !d \to \overline{a})) \end{array}$$

both synchronisations over c and e are possible, preparing the explicit substitutions $\langle y := N\rangle$ and $\langle x := L\rangle$, respectively. So reduction under '$\to_{\mathrm{XH}}$' is not deterministic, and therefore neither is reduction in the image of $\llbracket \cdot \rrbracket^{\mathrm{L}}_{\mathrm{J}} \cdot$.

We can make the following observations:

Remark 7.4 • As can be seen from the proofs of Lemma 6.7 and Thm 7.1, the synchronisations generated by the encoding only involve processes of the shape:

$$\begin{array}{l} x(u).!u{\to}\overline{a} \mid \overline{x}(w).\llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, w \\ \quad w(v).\overline{a}\,v \mid (\nu y b)\,(\llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, b \mid \overline{w}\langle y,b\rangle) \\ \qquad \overline{c}\langle x,b\rangle \mid c(v,d).(!\overline{v}(w).\llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, w \mid !d{\to}\overline{a}) \end{array}$$

so in particular, substitution is always well defined.

- A process that results from running the interpretation of a term M with free variable y will only able to input (on y) if it has a sub-process $y(u).!u{\to}\overline{a} \triangleq \llbracket y\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a$ that does not occur under guard. All other occurrences of y will appear inside $\llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, w$ in subprocesses like $!c(v,d).(!\overline{v}(w).\llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, w \mid !d{\to}\overline{c})$ or $\overline{x}(w).\llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, w$, so in particular appear under guard and are unavailable for synchronisation.

We can also show that no reductions are possible in $\llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a$ but those that correspond to reductions in M itself.

Theorem 7.5 (Operational Completeness for '$\to_{\mathrm{XH}}$') *Let $M \in \lambda\mu$.*

i) If $\llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a \to_{\pi} P$, then there exists Q, R, and N such that $P \approx_{\mathrm{R}} Q$, $Q \approx_{\mathrm{G}} R$, and $R \approx_{\mathrm{D}} \llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a$ (so $\llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a \to^{+}_{\pi}, \approx_{\mathrm{RGD}} \llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a$) and $M \to_{\mathrm{XH}} N$.

ii) If $\llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a \to^{+}_{\pi}, \approx_{\mathrm{RGD}} \llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a$ then there exists N' such that $\llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a \equiv \llbracket N'\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a$, and $M \to^{+}_{\mathrm{XH}} N'$.

Proof: i) By inspection of the cases of the proof for Thm 7.1, if $\llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a \to_{\pi} P$, there are only two cases where the reduction takes place in the interpreted term directly, and either:

$\llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a = (\nu c)\,((\nu x b)\,(\llbracket P\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, b \mid \overline{c}\langle x,b\rangle) \mid !c(v,d).(\llbracket v{:=}Q\rrbracket^{\mathrm{L}}_{\mathbb{J}} \mid !d{\to}\overline{a})) = \llbracket(\lambda x.P)\,Q\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a$: Then

$$\begin{array}{rl} & (\nu c)\,((\nu x b)\,(\llbracket P\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, b \mid \overline{c}\langle x,b\rangle) \mid !c(v,d).(\llbracket v{:=}Q\rrbracket^{\mathrm{L}}_{\mathbb{J}} \mid !d{\to}\overline{a})) \\ \to_{\pi} & (c) \quad (\nu c x b)\,(\llbracket P\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, b \mid \llbracket x{:=}Q\rrbracket^{\mathrm{L}}_{\mathbb{J}} \mid !b{\to}\overline{a} \mid !c(v,d).(\llbracket v{:=}Q\rrbracket^{\mathrm{L}}_{\mathbb{J}} \mid !d{\to}\overline{a})) \\ \equiv & (\nu x)\,((\nu b)\,(\llbracket P\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, b \mid !b{\to}\overline{a}) \mid \llbracket x{:=}Q\rrbracket^{\mathrm{L}}_{\mathbb{J}}) \mid (\nu c)\,(!c(v,d).(\llbracket v{:=}Q\rrbracket^{\mathrm{L}}_{\mathbb{J}} \mid !d{\to}\overline{a})) \\ \approx_{\mathrm{R}} & (\nu x)\,(\llbracket P\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a \mid \llbracket x{:=}Q\rrbracket^{\mathrm{L}}_{\mathbb{J}}) \mid (\nu c)\,(!c(v,d).(\llbracket v{:=}Q\rrbracket^{\mathrm{L}}_{\mathbb{J}} \mid !d{\to}\overline{a})) \\ \approx_{\mathrm{G}} & (\nu x)\,(\llbracket P\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a \mid \llbracket x{:=}Q\rrbracket^{\mathrm{L}}_{\mathbb{J}}) \qquad \triangleq \llbracket P\langle x{:=}Q\rangle\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a \end{array}$$

Notice that $(\lambda x.P)\,Q \to_{\mathrm{XH}} P\langle x{:=}Q\rangle$, and that congruence is needed to create a process in which the substitution can be isolated as garbage, which needs to be removed to obtain a process that is an interpreted term.

$\llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a = \llbracket x\boldsymbol{S}_0 M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n\langle x{:=}N\rangle\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a$: By Remark 6.6, we can move the substitutions to the outside and then, with $c_{n+1} = a$ and $\boldsymbol{S} = \boldsymbol{S}_0 \cdots \boldsymbol{S}_n$ (note that $!\overline{x}(w).\llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, w \in \llbracket \boldsymbol{S}\rrbracket^{\mathrm{L}}_{\mathbb{J}}$):

$$\begin{array}{rll} \llbracket x\boldsymbol{S}_0 M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a & \equiv (6.6) & (\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})\,(\llbracket x\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, c_1 \mid \overrightarrow{\llbracket c_i{:=}M_i{\cdot}c_{i+1}\rrbracket^{\mathrm{L}}_{\mathbb{J}}} \mid \llbracket \boldsymbol{S}\rrbracket^{\mathrm{L}}_{\mathbb{J}}) \\ & \triangleq & (\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})\,(x(u).!u{\to}\overline{c_1} \mid \overrightarrow{\llbracket c_i{:=}M_i{\cdot}c_{i+1}\rrbracket^{\mathrm{L}}_{\mathbb{J}}} \mid \llbracket \boldsymbol{S}\rrbracket^{\mathrm{L}}_{\mathbb{J}}) \\ & \equiv & (\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})\,(x(u).!u{\to}\overline{c_1} \mid \overrightarrow{\llbracket c_i{:=}M_i{\cdot}c_{i+1}\rrbracket^{\mathrm{L}}_{\mathbb{J}}} \mid \overline{x}(w).\llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, w \mid \llbracket \boldsymbol{S}\rrbracket^{\mathrm{L}}_{\mathbb{J}}) \\ & \to_{\pi} & (x) \quad (\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})\,((\nu w)\,(\llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, w \mid !w{\to}\overline{c_1}) \mid \overrightarrow{\llbracket c_i{:=}M_i{\cdot}c_{i+1}\rrbracket^{\mathrm{L}}_{\mathbb{J}}} \mid \llbracket \boldsymbol{S}\rrbracket^{\mathrm{L}}_{\mathbb{J}}) \\ & \approx_{\mathrm{R}} & (\nu\overrightarrow{y_n}\overrightarrow{\alpha_n})\,(\llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, c_1 \mid \overrightarrow{\llbracket c_i{:=}M_i{\cdot}c_{i+1}\rrbracket^{\mathrm{L}}_{\mathbb{J}}} \mid \llbracket \boldsymbol{S}\rrbracket^{\mathrm{L}}_{\mathbb{J}}) \\ & \triangleq & \llbracket N M_1 \cdots M_n \boldsymbol{S}\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a \;\equiv\; \llbracket N\boldsymbol{S}_0 M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a \end{array}$$

and $x\boldsymbol{S}_0 M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n\langle x{:=}N\rangle \to_{\mathrm{XH}} N\boldsymbol{S}_0 M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n\ \langle x{:=}N\rangle$.. The two steps where congruence is used are mainly for elegance, and not necessary for the proof.

Otherwise, reduction takes place inside an interpreted term, and the proof then follows by induction. By Remark 7.4, no other synchronisations are possible inside an interpreted term.

ii) Notice that, in part (*i*), if $\llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a \to_{\pi} P$, then there exists N such that $\llbracket M\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a \to^{*}_{\pi}, \approx_{\mathrm{RGD}} \llbracket N\rrbracket^{\mathrm{L}}_{\mathbb{J}}\, a$ and that also $M \to^{*}_{\mathrm{XH}} N$. The result follows by induction on the length of the reduction path, using the first part. Notice that renaming and garbage collection (that involves processes that are inactive with respect to synchronisation) can always be postponed until

at the end, and that the final step with '$\approx_{\mathrm{D}}$' is only needed to obtain the right syntactic presentation of N. Moreover, some of the intermediate processes might correspond to interpreted terms, but perhaps with the substitutions in the wrong place (as in the second case of proof of the first part, where the interpreted substitutions get moved for notational convenience); through congruence they can be brought to the right place, to correctly represent the result of reduction. □

In fact, it is easy to check that congruence plays no part in the proof of Thm 7.1 other than to simplify the presentation of the processes or to prepare garbage collection.

We can also show that standard reduction with explicit substitution, '$\to_{\mathrm{X}}$', is preserved under our encoding by weak bisimulation. Note that this result is stated for '$=_{\mathrm{X}}$', not '$=_{\mathrm{XH}}$', and that it does not show that the encoding of terms is related through reduction.

Theorem 7.6 *For all $M, N \in \lambda\mu\mathbf{x}$, if $M =_{\mathrm{X}} N$, then $\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a \approx \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a$.*

Proof: By induction on the definition of '$=_{\mathrm{X}}$'; we only show some of the cases that are different or not included in the proof of Thm 7.1.

$$
\begin{array}{lll}
x\langle x{:=}N\rangle \to N\colon \llbracket x\langle x{:=}N\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a & \triangleq\ (\nu x)\,(\llbracket x \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a \mid \llbracket x{:=}N \rrbracket^{\mathrm{L}}_{\mathrm{J}}) & \triangleq \\
\quad (\nu x)\,(x(u).!\,u_{\to}\overline{a} \mid !\,\overline{x}(w).\llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w) & \to_{\pi}\ (x) \quad (\nu w)\,(!\,w_{\to}\overline{a} \mid \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w) \mid (\nu x)\,(!\,\overline{x}(w).\llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w) & \approx_{\mathrm{G}} \\
\quad (\nu w)\,(!\,w_{\to}\overline{a} \mid \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w) & \approx_{\mathrm{R}}\ \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a &
\end{array}
$$

$$
\begin{array}{lll}
(PQ)\langle x{:=}N\rangle \to (P\langle x{:=}N\rangle)(Q\langle x{:=}N\rangle)\colon \llbracket (PQ)\langle x{:=}N\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a & \triangleq & \\
\quad (\nu x)\,((\nu c)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid !\,c(v,d).(\llbracket v{:=}Q \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d_{\to}\overline{a})) \mid !\,\overline{x}(w).\llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w) & \approx & (2.5{:}6) \\
\quad (\nu c)\,((\nu x)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid \llbracket x{:=}N \rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid (\nu x)\,(!\,c(v,d).(!\,\overline{v}(w).\llbracket Q \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w \mid d_{\to}\overline{a}) \mid \llbracket x{:=}N \rrbracket^{\mathrm{L}}_{\mathrm{J}})) & \approx & (2.5{:}8) \\
\quad (\nu c)\,((\nu x)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid \llbracket x{:=}N \rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid !\,c(v,d).((\nu x)\,(!\,\overline{v}(w).\llbracket Q \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w \mid \llbracket x{:=}N \rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid d_{\to}\overline{a})) & \approx & (2.5{:}9) \\
\quad (\nu c)\,((\nu x)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid \llbracket x{:=}N \rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid !\,c(v,d).(!\,\overline{v}(w).(\nu x)\,(\llbracket Q \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w \mid \llbracket x{:=}N \rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid !\,d_{\to}\overline{a})) & \triangleq & \\
\quad (\nu c)\,(\llbracket P\langle x{:=}N\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid !\,c(v,d).(!\,\overline{v}(w).\llbracket Q\langle x{:=}N\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w \mid !\,d_{\to}\overline{a})) & \triangleq & \\
\quad (\nu c)\,(\llbracket P\langle x{:=}N\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid !\,c(v,d).(\llbracket v{:=}Q\langle x{:=}N\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d_{\to}\overline{a})) & \triangleq & \\
\quad (\nu c)\,(\llbracket P\langle x{:=}N\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid \llbracket c{:=}(Q\langle x{:=}N\rangle)\cdot a \rrbracket^{\mathrm{L}}_{\mathrm{J}}) & \triangleq & \\
\quad \llbracket (P\langle x{:=}N\rangle)(Q\langle x{:=}N\rangle) \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a & &
\end{array}
$$

$$
\begin{array}{lll}
(PQ)\langle \alpha{:=}N\cdot\gamma\rangle \to (P\langle \alpha{:=}N\cdot\gamma\rangle)\ (Q\langle \alpha{:=}N\cdot\gamma\rangle)\colon \llbracket (PQ)\langle \alpha{:=}N\cdot\gamma\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a & \triangleq & \\
\quad (\nu\alpha)\,((\nu c)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid \llbracket c{:=}Q\cdot a \rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid \llbracket \alpha{:=}N\cdot\gamma \rrbracket^{\mathrm{L}}_{\mathrm{J}}) & \triangleq & \\
\quad (\nu\alpha)\,((\nu c)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid !\,c(v,d).(\llbracket v{:=}Q \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d_{\to}\overline{a})) \mid !\,\alpha(v,d).(!\,\overline{v}(w).\llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w \mid !\,d_{\to}\overline{\gamma})) & \approx & (2.5{:}1) \\
\quad (\nu c)\,((\nu\alpha)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid \llbracket \alpha{:=}N\cdot\gamma \rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid (\nu\alpha)\,(!\,c(v,d).(\llbracket v{:=}Q \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d_{\to}\overline{a}) \mid \llbracket \alpha{:=}N\cdot\gamma \rrbracket^{\mathrm{L}}_{\mathrm{J}})) & \approx & (2.5{:}4) \\
\quad (\nu c)\,((\nu\alpha)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid \llbracket \alpha{:=}N\cdot\gamma \rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid !\,c(v,d).((\nu\alpha)\,(!\,\overline{v}(w).\llbracket Q \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w \mid \llbracket \alpha{:=}N\cdot\gamma \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d_{\to}\overline{a}))) & \approx & (2.5{:}5) \\
\quad (\nu c)\,((\nu\alpha)\,(\llbracket P \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid \llbracket \alpha{:=}N\cdot\gamma \rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid !\,c(v,d).(!\,\overline{v}(w).(\nu\alpha)\,(\llbracket Q \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w \mid \llbracket \alpha{:=}N\cdot\gamma \rrbracket^{\mathrm{L}}_{\mathrm{J}}) \mid !\,d_{\to}\overline{a})) & \triangleq & \\
\quad (\nu c)\,(\llbracket P\langle \alpha{:=}N\cdot\gamma\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid !\,c(v,d).(!\,\overline{v}(w).\llbracket Q\langle \alpha{:=}N\cdot\gamma\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w \mid !\,d_{\to}\overline{a})) & \triangleq & \\
\quad (\nu c)\,(\llbracket P\langle \alpha{:=}N\cdot\gamma\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid !\,c(v,d).(\llbracket v{:=}Q\langle \alpha{:=}N\cdot\gamma\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d_{\to}\overline{a})) & \triangleq & \\
\quad (\nu c)\,(\llbracket P\langle \alpha{:=}N\cdot\gamma\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid \llbracket c{:=}(Q\langle \alpha{:=}N\cdot\gamma\rangle)\cdot a \rrbracket^{\mathrm{L}}_{\mathrm{J}}) & \triangleq & \\
\quad \llbracket (P\langle \alpha{:=}N\cdot\gamma\rangle)(Q\langle \alpha{:=}N\cdot\gamma\rangle) \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a & &
\end{array}
$$

$$
\begin{array}{lll}
M \to N \Rightarrow LM \to LN\colon \llbracket LM \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a \triangleq (\nu c)\,(\llbracket L \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid c(v,d).(\llbracket v{:=}M \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d_{\to}\overline{a}) \mid \llbracket c{:=}M\cdot a \rrbracket^{\mathrm{L}}_{\mathrm{J}}) & \approx & (IH) \\
\quad (\nu c)\,(\llbracket L \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, c \mid c(v,d).(\llbracket v{:=}N \rrbracket^{\mathrm{L}}_{\mathrm{J}} \mid !\,d_{\to}\overline{a}) \mid \llbracket c{:=}N\cdot a \rrbracket^{\mathrm{L}}_{\mathrm{J}}) \triangleq \llbracket LN \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a & &
\end{array}
$$

$$
\begin{array}{lll}
M \to N \Rightarrow L\langle x{:=}M\rangle \to L\langle x{:=}N\rangle\colon \llbracket L\langle x{:=}M\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a \triangleq (\nu x)\,(\llbracket L \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a \mid !\,\overline{x}(w).\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w) & \approx & (IH) \\
\quad (\nu x)\,(\llbracket L \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a \mid !\,\overline{x}(w).\llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, w) \triangleq \llbracket L\langle x{:=}N\rangle \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a & &
\end{array}
$$

The steps to a reflexive, transitive closure and equivalence relation follow directly from the fact that '$\approx$' is a congruence, as in the last two parts shown above. □

Notice that, for the inductive cases, we apply induction to a process occurring under guard, so need that '$\approx$' is a congruence, so Lemma 2.5 alone is no longer sufficient.

Now the following is an immediate consequence:

Theorem 7.7 (Semantics) *For all $M, N \in \lambda\mu$, if $M =_{\beta\mu} N$, then $\llbracket M \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a \approx \llbracket N \rrbracket^{\mathrm{L}}_{\mathrm{J}}\, a$.*

Proof: By induction on the definition of '$=_{\beta\mu}$'. The case $M \to^{*}_{\beta\mu} N$ follows from the fact that

then, by Proposition 4.3, also $M \to_{\mathrm{X}}^{*} N$, so by Thm 7.6, we have $\llbracket M \rrbracket_{\mathrm{L}}^{a} \approx \llbracket N \rrbracket_{\mathrm{L}}^{a}$. The steps to an equivalence relation follow directly from '$\approx$'. $\square$

Notice that it is clear that we cannot prove the exact reversal of this result, since terms without head-normal form are all interpreted by a process that is weakly bisimilar to 0 (see also Lemma 8.7), but are not all related through '$=_{\beta\mu}$'. However, similar to [51, 52], using a notion of weak equivalence we can deal with the reverse part and will do so in the last sections of this paper.

We can show that interpretation of terms in $\to_{\mathrm{XH}}$-normal form are in normal form as well.

Lemma 7.8 If $\boldsymbol{N}$ is a $\to_{\mathrm{XH}}$-normal form, then $\llbracket \boldsymbol{N} \rrbracket_{\mathrm{L}}^{a}$ is irreducible.

Proof: By induction on the structure of terms in $\to_{\mathrm{XH}}$-normal form.

$\boldsymbol{N} = xM_1\boldsymbol{S}_1 \cdots M_n\boldsymbol{S}_n$ $(n \geq 0)$: $\llbracket xM_1\boldsymbol{S}_1 \cdots M_n\boldsymbol{S}_n \rrbracket_{\mathrm{L}}^{a} \triangleq, \equiv$ (6.6)
$(\nu\overrightarrow{y_n}\overrightarrow{\alpha_n}) \cdots (\nu\overrightarrow{y_1}\overrightarrow{\alpha_1})\,(\nu\vec{c})\,(\llbracket x \rrbracket_{\mathrm{L}}^{c_1} \mid \llbracket c_1 := (M_1)\cdot c_2 \rrbracket_{\mathrm{L}} \mid \cdots \mid \llbracket c_n := (M_n)\cdot a \rrbracket_{\mathrm{L}} \mid \llbracket \boldsymbol{S}_1 \rrbracket_{\mathrm{L}} \mid \cdots \mid \llbracket \boldsymbol{S}_n \rrbracket_{\mathrm{L}})$

Since
$$\begin{array}{rcl} \llbracket x \rrbracket_{\mathrm{L}}^{c_1} & \triangleq & x(u).!u \to \overline{c_1} \\ \llbracket c_i := (M_i)\cdot c_{i+1} \rrbracket_{\mathrm{L}} & \triangleq & !c_i(v,d).(!\overline{v}(w).\llbracket M_i \rrbracket_{\mathrm{L}}^{w} \mid !d \to \overline{c_{i+1}}) \\ \llbracket y_j := N_j \rrbracket_{\mathrm{L}} \in \boldsymbol{S}_i & \triangleq & \overline{y_j}(w).\llbracket N_j \rrbracket_{\mathrm{L}}^{w} \\ \llbracket \alpha_j := P_j\cdot\gamma_j \rrbracket_{\mathrm{L}} \in \boldsymbol{S}_i & \triangleq & !\alpha_j(v,d).(!\overline{v}(w).\llbracket P_j \rrbracket_{\mathrm{L}}^{w} \mid !d \to \overline{\gamma_j}) \end{array}$$
all $\llbracket M_i \rrbracket_{\mathrm{L}}^{w}$, $\llbracket N_j \rrbracket_{\mathrm{L}}^{w}$, and $\llbracket P_j \rrbracket_{\mathrm{L}}^{w}$ appear under input, so no synchronisation inside one of those is possible; since all c_i are fresh, all are different from x and no synchronisation is possible over any of the c_i. Since x does not appear in any of the $\boldsymbol{S}_i$, also no synchronisation over x is possible. So this process is in normal form.

$\boldsymbol{N} = \lambda x.\boldsymbol{N}'$: Then $\llbracket \lambda x.\boldsymbol{N}' \rrbracket_{\mathrm{L}}^{a} \triangleq (\nu xb)\,(\llbracket \boldsymbol{N}' \rrbracket_{\mathrm{L}}^{b} \mid \overline{a}\langle x,b\rangle)$, and, by induction, $\llbracket \boldsymbol{N}' \rrbracket_{\mathrm{L}}^{b}$ is in normal form; since b is fresh and $a \notin \llbracket \boldsymbol{N}' \rrbracket_{\mathrm{L}}^{b}$, that process does not input over a, so $\llbracket \lambda x.\boldsymbol{N}' \rrbracket_{\mathrm{L}}^{a}$ is in normal form.

$\boldsymbol{N} = \mu\alpha.[\beta]\boldsymbol{N}'$ $(\alpha \neq \beta \vee \alpha \in \boldsymbol{N}', \boldsymbol{N}' \neq \mu\gamma.[\delta]\boldsymbol{N}'')$: Then $\llbracket \mu\alpha.[\beta]\boldsymbol{N}' \rrbracket_{\mathrm{L}}^{a} \triangleq \llbracket \boldsymbol{N}' \rrbracket_{\mathrm{L}}^{\beta}\{a/\alpha\}$; this case follows immediately by induction. $\square$

Notice that $\llbracket \mu\alpha.[\beta]\mu\gamma.[\delta]\boldsymbol{N} \rrbracket_{\mathrm{L}}^{a} = \llbracket \boldsymbol{N} \rrbracket_{\mathrm{L}}^{\delta}\{\beta/\gamma\}\{a/\alpha\}$, which is in normal form, so some reducible terms in $\lambda\mu\mathbf{x}$ are mapped to processes in normal form; this does not contradict the above result, of course.

We can now show the following termination results:

Theorem 7.9 (Termination) *i) If $M \to_{\mathrm{XH}}^{nf} N$, then $\llbracket M \rrbracket_{\mathrm{L}}^{a} \Downarrow_{\pi}$.*
ii) If $M \to_{\beta\mu}^{hnf} N$, then $\llbracket M \rrbracket_{\mathrm{L}}^{a} \Downarrow_{\pi}$.

Proof: i) By Lemma 7.8, if N is in explicit head-normal from, then $\llbracket N \rrbracket_{\mathrm{L}}^{a}$ is in normal form. By Thm 7.5, there exists P such that $\llbracket M \rrbracket_{\mathrm{L}}^{a} \to_{\pi}^{+} P$ with $P \approx_{\mathrm{RGD}} \llbracket N \rrbracket_{\mathrm{L}}^{a}$. It might be that in the '$\approx_{\mathrm{R}}$'-part, synchronisations take place; we can add those to the '$\to_{\pi}^{+}$' steps and can assume that we have $\llbracket M \rrbracket_{\mathrm{L}}^{a} \to_{\pi}^{+} P$ with $P \approx_{\mathrm{RGD}} \llbracket N \rrbracket_{\mathrm{L}}^{a}$, and the latter does not involve synchronisations, so normal forms are preserved. So P is weakly bisimilar to a process in normal form, and in establishing that relation, no synchronisations are needed; remark that, in the proof of Thm 7.5, '$\approx_{\mathrm{G}}$' only removes irreducible processes (in normal form). This implies that P is in normal form.

ii) By Proposition 1.8, there exists L in HNF such that $M \to_{\mathrm{H}}^{nf} L$; by Lemma 5.4, there exists $\boldsymbol{N}$ such that $M \to_{\mathrm{XH}}^{nf} \boldsymbol{N}$; by the previous part, $\llbracket M \rrbracket_{\mathrm{L}}^{a} \Downarrow_{\pi}$. $\square$

Notice also that this result is stronger than the formulation of the termination result for Milner's interpretation in [52] (or any other), since it models reduction to head-normal form, not just lazy normal form.

Since terms that have a normal form have a head-normal form as well, Thm 7.9 immediately leads to:

$$
\begin{array}{rcl}
\llbracket \Delta\Delta \rrbracket^{\mathrm{L}}_{\rrbracket} a & \triangleq & (\nu c)\,((\nu xb)\,(\llbracket xx \rrbracket^{\mathrm{L}}_{\rrbracket} b \mid \overline{c}\langle x,b\rangle) \mid !c(v,d).(\llbracket v := \Delta \rrbracket^{\mathrm{L}}_{\rrbracket} \mid !d{\rightarrow}\overline{a})) \\
\rightarrow_{\pi}(c) & & (\nu x)\,(\llbracket xx \rrbracket^{\mathrm{L}}_{\rrbracket} a \mid \llbracket x := \Delta \rrbracket^{\mathrm{L}}_{\rrbracket}) \mid (\nu c)\,(\llbracket c := \Delta\cdot a \rrbracket^{\mathrm{L}}_{\rrbracket}) \\
\approx_{\mathrm{G}} & & (\nu x)\,(\llbracket xx \rrbracket^{\mathrm{L}}_{\rrbracket} a \mid \llbracket x := \Delta \rrbracket^{\mathrm{L}}_{\rrbracket}) \\
\triangleq & & \llbracket xx\,\langle x := \Delta\rangle \rrbracket^{\mathrm{L}}_{\rrbracket} a \\
\triangleq & & (\nu x)\,((\nu c)\,(x(u).!u{\rightarrow}\overline{c} \mid !c(v,d).(\llbracket v := x \rrbracket^{\mathrm{L}}_{\rrbracket} \mid !d{\rightarrow}\overline{a})) \mid !\overline{x}(w).(\nu yb)\,(\llbracket yy \rrbracket^{\mathrm{L}}_{\rrbracket} b \mid \overline{w}\langle y,b\rangle)) \\
\rightarrow_{\pi}(x,w) & & (\nu x)\,((\nu c)\,((\nu yb)\,(\llbracket yy \rrbracket^{\mathrm{L}}_{\rrbracket} b \mid \overline{c}\langle y,b\rangle) \mid !c(v,d).(\llbracket v := x \rrbracket^{\mathrm{L}}_{\rrbracket} \mid !d{\rightarrow}\overline{a})) \mid \llbracket x := \Delta \rrbracket^{\mathrm{L}}_{\rrbracket}) \\
\triangleq & & \llbracket (\lambda y.yy)\,x\,\langle x := \Delta\rangle \rrbracket^{\mathrm{L}}_{\rrbracket} a \\
\rightarrow_{\pi}(c) & & (\nu x)\,((\nu yb)\,(\llbracket yy \rrbracket^{\mathrm{L}}_{\rrbracket} b \mid \llbracket y := x \rrbracket^{\mathrm{L}}_{\rrbracket} \mid !d{\rightarrow}\overline{a} \mid \llbracket x := \Delta \rrbracket^{\mathrm{L}}_{\rrbracket}) \mid (\nu c)\,(\llbracket c := x\cdot a \rrbracket^{\mathrm{L}}_{\rrbracket})) \\
\approx_{\mathrm{G}}, \triangleq & & (\nu x)\,((\nu y)\,((\nu c)\,(y(u).!u{\rightarrow}\overline{c} \mid \llbracket c := y\cdot a \rrbracket^{\mathrm{L}}_{\rrbracket}) \mid !\overline{y}(w).\llbracket x \rrbracket^{\mathrm{L}}_{\rrbracket} w) \mid \llbracket x := \Delta \rrbracket^{\mathrm{L}}_{\rrbracket}) \\
\triangleq & & \llbracket yy\,\langle y := x\rangle\,\langle x := \Delta\rangle \rrbracket^{\mathrm{L}}_{\rrbracket} a \\
\rightarrow_{\pi}(y) & & (\nu x)\,((\nu y)\,((\nu cw)\,(!w{\rightarrow}\overline{c} \mid \llbracket c := y\cdot a \rrbracket^{\mathrm{L}}_{\rrbracket} \mid \llbracket x \rrbracket^{\mathrm{L}}_{\rrbracket} w) \mid \llbracket y := x \rrbracket^{\mathrm{L}}_{\rrbracket}) \mid \llbracket x := \Delta \rrbracket^{\mathrm{L}}_{\rrbracket}) \\
\approx_{\mathrm{R}} & & (\nu x)\,((\nu y)\,((\nu c)\,(\llbracket x \rrbracket^{\mathrm{L}}_{\rrbracket} c \mid \llbracket c := y\cdot a \rrbracket^{\mathrm{L}}_{\rrbracket}) \mid \llbracket y := x \rrbracket^{\mathrm{L}}_{\rrbracket}) \mid \llbracket x := \Delta \rrbracket^{\mathrm{L}}_{\rrbracket}) \\
\triangleq & & \llbracket xy\,\langle y := x\rangle\,\langle x := \Delta\rangle \rrbracket^{\mathrm{L}}_{\rrbracket} a \\
\rightarrow^{*}_{\pi}, \approx_{\mathrm{G}} & & (\nu x)\,((\nu y)\,((\nu c)\,(\llbracket \lambda z.zz \rrbracket^{\mathrm{L}}_{\rrbracket} c \mid \llbracket c := y\cdot a \rrbracket^{\mathrm{L}}_{\rrbracket}) \mid \llbracket y := x \rrbracket^{\mathrm{L}}_{\rrbracket}) \mid \llbracket x := \Delta \rrbracket^{\mathrm{L}}_{\rrbracket}) \\
\triangleq & & \llbracket (\lambda z.zz)\,y\,\langle y := x\rangle\,\langle x := \Delta\rangle \rrbracket^{\mathrm{L}}_{\rrbracket} a \\
\rightarrow^{*}_{\pi}, \approx_{\mathrm{RG}} & & (\nu x)\,((\nu y)\,((\nu z)\,(\llbracket zz \rrbracket^{\mathrm{L}}_{\rrbracket} a \mid \llbracket z := y \rrbracket^{\mathrm{L}}_{\rrbracket}) \mid \llbracket y := x \rrbracket^{\mathrm{L}}_{\rrbracket}) \mid \llbracket x := \Delta \rrbracket^{\mathrm{L}}_{\rrbracket}) \\
\triangleq & & \llbracket zz\,\langle z := y\rangle\,\langle y := x\rangle\,\langle x := \Delta\rangle \rrbracket^{\mathrm{L}}_{\rrbracket} a \qquad \ldots
\end{array}
$$

Figure 4: Running $\llbracket (\lambda x.xx)\,(\lambda x.xx) \rrbracket^{\mathrm{L}}_{\rrbracket} a$

Corollary 7.10 *If $M{\Downarrow_{\beta\mu}}$, then $\llbracket M \rrbracket^{\mathrm{L}}_{\rrbracket} a{\Downarrow_{\pi}}$.*

8 Weak reduction for $\lambda\mu$ and $\lambda\mu$x

It seems widely accepted that bisimilarity-like equivalences have become the standard when studying interpretations of λ-calculi into the π-calculus. This creates a point of concern with respect to full abstraction. Since $\Delta\Delta$ and $\Omega\Omega$ (where $\Omega = \lambda y.yyy$; we will use Ω again below) are closed terms that do not interact with any context, they are contextually equivalent; any well-defined interpretation of these terms into the π-calculus, be it input based or output based, will therefore map those to processes that are weakly bisimilar to 0, and therefore to weakly bisimilar processes. Abstraction, on the other hand, enables interaction with a context, and therefore the interpretation of $\lambda z.\Delta\Delta$ will *not* be weakly bisimilar to 0. However, in any standard model of β-reduction of the λ-calculus, the terms $\Delta\Delta$ and $\lambda z.\Delta\Delta$ are equated since both are *meaningless* (they are both *unsolvable* [57, 58]). We therefore cannot hope to model normal $\beta\mu$-equality in the π-calculus in a fully-abstract way; rather, we need to consider a notion of reduction that considers *all* abstractions meaningful; therefore, the only kind of reduction on λ-calculi that can naturally be encoded into the π-calculus in a fully-abstract way is *weak* reduction.

Example 8.1 Consider the reduction of $\Delta\Delta$ that was given in Example 5.5; by Thm 7.1, we have that $\llbracket \Delta\Delta \rrbracket^{\mathrm{L}}_{\rrbracket} a \approx \llbracket zz\,\langle z := y\rangle\,\langle y := x\rangle\,\langle x := \Delta\rangle \rrbracket^{\mathrm{L}}_{\rrbracket} a$ as shown in Fig. 4; notice that the individual steps of the reduction in '$\rightarrow_{\mathrm{XH}}$' in Example 5.5 are respected in Fig. 4 This reduction illustrates that the interpretation of $\Delta\Delta$ reduces without creating output over a – that name always occurs inside a sub-process of the shape

$$\llbracket c := y\cdot a \rrbracket^{\mathrm{L}}_{\rrbracket} \;\triangleq\; !c(v,d).(\llbracket v := y \rrbracket^{\mathrm{L}}_{\rrbracket} \mid !d{\rightarrow}\overline{a})$$

and does not input, since all occurrences of variables are always bound, so $\llbracket \Delta\Delta \rrbracket^{\mathrm{L}}_{\rrbracket} a$ is weakly bisimilar to 0 (see also Lemma 8.7). Therefore,

$$
\begin{array}{rcl}
\llbracket \lambda z.\Delta\Delta \rrbracket^{\mathrm{L}}_{\rrbracket} a \;\triangleq\; (\nu zb)\,(\llbracket \Delta\Delta \rrbracket^{\mathrm{L}}_{\rrbracket} b \mid \overline{a}\langle z,b\rangle) & \approx & (\nu zb)\,(0 \mid \overline{a}\langle z,b\rangle) \;\approx \\
(\nu zb)\,(\llbracket \Omega\Omega \rrbracket^{\mathrm{L}}_{\rrbracket} b \mid \overline{a}\langle z,b\rangle) \;\triangleq\; \llbracket \lambda z.\Omega\Omega \rrbracket^{\mathrm{L}}_{\rrbracket} a & & \\
\end{array}
$$

So, for full abstraction, we are forced to consider $\lambda z.\Delta\Delta$ and $\lambda z.\Omega\Omega$ equivalent and both different from $\Delta\Delta$, and therefore, we need to consider *weak* equivalences on terms.

We will now introduce the correct notions in the various version of $\lambda\mu$ we consider here.

Definition 8.2 (Weak reduction) We define the notion '$\to_{w\beta\mu}$' of *weak $\beta\mu$-reduction* as '$\to_{\beta\mu}$ in Def. 1.4, the notion '$\to_{w\text{H}}$' of *weak head reduction* on $\lambda\mu$ as '$\to_{\text{H}}$' in Def. 1.7, and the notion '$\to_{w\text{xH}}$' of *weak explicit head reduction* as '$\to_{\text{xH}}$' in Def. 5.1, in each case by (also) eliminating the rule:

$$M \to N \;\Rightarrow\; \lambda x.M \to \lambda x.N$$

Notice that we have chosen not to use the moniker 'lazy' but rather call these notions 'weak'; for lazy reduction the natural choice would have been to also eliminate the rule $(\lambda\boldsymbol{S})$: $(\lambda y.M)\boldsymbol{S} \to \lambda y.(M\boldsymbol{S})$.

Since in our interpretation abstraction is modelled using an asynchronous output, however,

$$\begin{array}{lllll} \llbracket(\lambda y.M)\boldsymbol{S}^{\text{L}}\rrbracket\, a & \triangleq & (\nu\vec{x})\,(\llbracket\lambda y.M^{\text{L}}\rrbracket\, a \mid \llbracket\boldsymbol{S}^{\text{L}}\rrbracket) & \triangleq & (\nu\vec{x})\,((\nu yb)\,(\llbracket M^{\text{L}}\rrbracket\, b \mid \overline{a}\langle y,b\rangle) \mid \llbracket\boldsymbol{S}^{\text{L}}\rrbracket) \;\equiv \\ & & (\nu yb)\,((\nu\vec{x})\,(\llbracket M^{\text{L}}\rrbracket\, b \mid \llbracket\boldsymbol{S}^{\text{L}}\rrbracket) \mid \overline{a}\langle x,b\rangle) & \triangleq & (\nu yb)\,(\llbracket M\boldsymbol{S}^{\text{L}}\rrbracket\, b \mid \overline{a}\langle x,b\rangle) \;\triangleq \\ & & \llbracket\lambda y.(M\boldsymbol{S})^{\text{L}}\rrbracket\, a & & \end{array}$$

the step $\lambda\boldsymbol{S}$ is modelled under our interpretation regardless; allowing it does not alter the structure of the proofs much, apart from the fact that then we would have $(\lambda x.M)\boldsymbol{S}$ as a term in explicit weak head normal form, rather than $\lambda x.M$ as we have below, which has a knock-on effect on a number of definitions below. For a notion of explicit lazy reduction for the λ-calculus, see Def. A.2.

We can show the following property.

Lemma 8.3 i) Let $M, N \in \lambda\mu$; then $M \to^{nf}_{w\text{H}} N$ if and only if there exists $N' \in \lambda\mu\mathbf{x}$ such that $M \to^{nf}_{w\text{xH}} N'$, and $N' \to^{nf}_{:=} N$.

ii) For $M, N \in \lambda\mu\mathbf{x}$: if $M \to^{*}_{w\text{xH}} N$, and $M \to^{nf}_{:=} M'$ and $N \to^{nf}_{:=} N'$, then $M' \to^{*}_{w\text{H}} N'$.

Proof: Straightforward, similar to Lemma 5.4. □

We define the notion of weak head-normal forms, the normal forms with respect to weak head-reduction:

Definition 8.4 (Weak head-normal forms for $\lambda\mu$ and $\lambda\mu\mathbf{x}$) *i*) The $\lambda\mu$ *weak head-normal forms* (whnf) are defined through the grammar:

$$\begin{array}{rcll} \boldsymbol{H}_w & ::= & \lambda x.M & (M \in \lambda\mu) \\ & \mid & xM_1\cdots M_n & (n \geq 0,\ \forall i \in \underline{n}\ (M_i \in \lambda\mu)) \\ & \mid & \mu\alpha.[\beta]\boldsymbol{H}_w & (\alpha \neq \beta \text{ or } \alpha \in \boldsymbol{H}_w, \text{ and } \boldsymbol{H}_w \neq \mu\gamma.[\delta]\boldsymbol{H}'_w) \end{array}$$

We say that M *has a* whnf if there exists $\boldsymbol{H}_w$ such that $M \to^{*}_{w\text{H}} \boldsymbol{H}_w$.

ii) The $\lambda\mu\mathbf{x}$ *weak explicit head-normal forms* (wxhnf) are defined through the grammar:

$$\begin{array}{rcll} \boldsymbol{H}_{w\text{x}} & ::= & \lambda x.M & (M \in \lambda\mu\mathbf{x}) \\ & \mid & xM_1\boldsymbol{S}_1\cdots M_n\boldsymbol{S}_n & (n \geq 0,\ \forall i \in \underline{n}\ [x \notin \boldsymbol{S}_i,\ M_i \in \lambda\mu\mathbf{x}, \\ & & & \quad c \in \boldsymbol{S}_i \Rightarrow c \in fv(xM_1\boldsymbol{S}_1\cdots M_i\boldsymbol{S}_i)]) \\ & \mid & \mu\alpha.[\beta]\boldsymbol{H}_{w\text{x}} & (\alpha \neq \beta \textit{ or } \alpha \in \boldsymbol{H}_{w\text{x}}, \textit{ and } \boldsymbol{H}_{w\text{x}} \neq \mu\gamma.[\delta]\boldsymbol{H}'_{w\text{x}}) \end{array}$$

We say that $M \in \lambda\mu\mathbf{x}$ *has a* wxhnf if there exists $\boldsymbol{H}_{w\text{x}}$ such that $M \to^{*}_{w\text{xH}} \boldsymbol{H}_w$.

As before, it is easy to verify that whnfs are the normal forms of weak head reduction.

The main difference between hnfs and whnfs is in the case of abstraction: where the definition of hnf only allows for the abstraction over a hnf, for whnfs the body can be any term. For example, both $\lambda z.\Delta\Delta$ and $\lambda z.\Omega\Omega$ are in whnf, but not in hnf. In fact, both terms have no hnf.

Since '$\to_{w\text{xH}}$' $\subseteq$ '$\to_{\text{xH}}$', we can show the equivalent of Proposition 1.8 and Thm 7.2 also for *weak explicit head reduction*.

Proposition 8.5 *If $M \to^*_{\beta\mu} N$ with N in* WHNF*, then there exists $\boldsymbol{H}_w$ such that $M \to^{nf}_{w\text{H}} \boldsymbol{H}_w$ and $\boldsymbol{H}_w \to^*_{\beta\mu} N$ without using '$\to_{w\text{H}}$'.*

Theorem 8.6 *i) If $M \to^*_{w\text{XH}} N$, then $\llbracket M \rrbracket^{\text{L}}_{\rrbracket} a \approx \llbracket N \rrbracket^{\text{L}}_{\rrbracket} a$.*
ii) If $M \Uparrow_{w\text{XH}}$, then $\llbracket M \rrbracket^{\text{L}}_{\rrbracket} a \Uparrow_{\pi}$.

We can show that the interpretation of a term without WHNF gives a process that is weakly bisimilar to $\mathtt{0}$.

Lemma 8.7 *If M has no* WXHNF *(so M also has no* WHNF*), then $\llbracket M \rrbracket^{\text{L}}_{\rrbracket} a \approx \mathtt{0}$.*

Proof: We will show that the interpretation of a term with a weak explicit head-redex has no input or output; since terms without WXHNF can only reduce (by contracting the head-redex) to a term without WXHNF, the interpretation of a term without a WXHNF will never input or output, and therefore be weakly equivalent to $\mathtt{0}$.

If M has no WXHNF, then M has no leading abstractions and all terms generated by reduction have a weak explicit head redex. If $M = \mu\alpha.[\beta]N$ and $\llbracket N \rrbracket^{\text{L}}_{\rrbracket} \beta\{a/\alpha\} \approx \mathtt{0}$, then also $\llbracket M \rrbracket^{\text{L}}_{\rrbracket} a \approx \mathtt{0}$; therefore we can assume M itself does not start with a context switch.

Let $M = R\boldsymbol{S}_0 P_1 \boldsymbol{S}_1 \cdots P_n \boldsymbol{S}_n$ with $n \geq 0$ and each $\boldsymbol{S}_i$ possibly empty, and let $\boldsymbol{S} = \boldsymbol{S}_0 \boldsymbol{S}_1 \cdots \boldsymbol{S}_n$, and:

$$\begin{array}{rcl}
\llbracket M \rrbracket^{\text{L}}_{\rrbracket} a & \equiv(6.6) & (\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,(\llbracket R \rrbracket^{\text{L}}_{\rrbracket} c_1 \mid \overrightarrow{\llbracket c_i := P_i \cdot c_{i+1} \rrbracket^{\text{L}}_{\rrbracket}}) \mid \llbracket \boldsymbol{S} \rrbracket^{\text{L}}_{\rrbracket}) \\
\llbracket c_i := P_i \cdot c_{i+1} \rrbracket^{\text{L}}_{\rrbracket} & \triangleq & !\,c_i(v,d).(!\,\overline{v}(w).\llbracket P_i \rrbracket^{\text{L}}_{\rrbracket} w \mid !\,d \rightarrow \overline{c_{i+1}}) \\
\llbracket \boldsymbol{S} \rrbracket^{\text{L}}_{\rrbracket} & \equiv & \overrightarrow{!\,\overline{y_j}(w).\llbracket Q_j \rrbracket^{\text{L}}_{\rrbracket} w} \mid \overrightarrow{!\,\alpha_k(v,d).(\llbracket v := T_k \rrbracket^{\text{L}}_{\rrbracket} \mid !\,d \rightarrow \overline{\beta_k})}
\end{array}$$

where $c_{n+1} = a$. Notice that all inputs and outputs in the interpretation of the substitutions are over bound names or under guard and that the only part of this process which might input our output is $\llbracket R \rrbracket^{\text{L}}_{\rrbracket} c_1$. We reason by coinduction on the infinite reduction path and distinguish the possibilities for the first reduction step. We show the more interesting cases:

(β): Then $R = \lambda x.K$, $n \geq 1$, and M contracts to $K\langle x := P_1\rangle \boldsymbol{S}_2 P_2 \boldsymbol{S}_1 \cdots P_n \boldsymbol{S}_n$. By coinduction, the process

$$\begin{array}{l}
\llbracket K\langle x := P_1\rangle \boldsymbol{S}_2 P_2 \boldsymbol{S}_1 \cdots P_n \boldsymbol{S}_n \rrbracket^{\text{L}}_{\rrbracket} a \\
\quad \triangleq, \equiv \;\; (\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,((\nu x)\,(\llbracket K \rrbracket^{\text{L}}_{\rrbracket} c_1 \mid !\,\overline{x}(w).\llbracket P_1 \rrbracket^{\text{L}}_{\rrbracket} w) \mid \overrightarrow{\llbracket c_i := P_i \cdot c_{i+1} \rrbracket^{\text{L}}_{\rrbracket}}) \mid \llbracket \boldsymbol{S} \rrbracket^{\text{L}}_{\rrbracket})
\end{array}$$

does not exhibit inputs or outputs, so, in particular, $(\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,(\llbracket K \rrbracket^{\text{L}}_{\rrbracket} c_1))$ does not (notice that $c_1 \in \vec{c}$ is bound). Then neither does $(\nu\vec{\alpha}\vec{y}\vec{c}xb)\,(\llbracket K \rrbracket^{\text{L}}_{\rrbracket} b)$ in $\llbracket M \rrbracket^{\text{L}}_{\rrbracket} a$.

(μ_p): Then $R = \mu\alpha.[\alpha]K$, and $n \geq 1$, and M contracts to $(\mu\gamma.[\gamma]K\langle\alpha := P_1 \cdot \gamma\rangle P_1)\boldsymbol{S}_2 P_2 \boldsymbol{S}_1 \cdots P_n \boldsymbol{S}_n$. By coinduction, the process

$$\begin{array}{l}
\llbracket (\mu\gamma.[\gamma \rrbracket^{\text{L}}_{\rrbracket} K\langle\alpha := P_1 \cdot \gamma\rangle P_1)\boldsymbol{S}_2 P_2 \boldsymbol{S}_1 \cdots P_n \boldsymbol{S}_n]a \\
\quad \triangleq, \equiv \;\; (\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,((\nu c')\,((\nu\alpha)\,(\llbracket K \rrbracket^{\text{L}}_{\rrbracket} c' \mid \llbracket \alpha := P_1 \cdot \gamma \rrbracket^{\text{L}}_{\rrbracket}) \mid \llbracket c' := P_1 \cdot \gamma \rrbracket^{\text{L}}_{\rrbracket})\,\{c_1/\gamma\} \mid \overrightarrow{\llbracket c_i := P_i \cdot c_{i+1} \rrbracket^{\text{L}}_{\rrbracket}}) \mid \llbracket \boldsymbol{S} \rrbracket^{\text{L}}_{\rrbracket}) \\
\quad = \;\; (\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,((\nu c')\,((\nu\alpha)\,(\llbracket K \rrbracket^{\text{L}}_{\rrbracket} c' \mid \llbracket \alpha := P_1 \cdot c_1 \rrbracket^{\text{L}}_{\rrbracket}) \mid \llbracket c' := P_1 \cdot c_1 \rrbracket^{\text{L}}_{\rrbracket}) \mid \overrightarrow{\llbracket c_i := P_i \cdot c_{i+1} \rrbracket^{\text{L}}_{\rrbracket}}) \mid \llbracket \boldsymbol{S} \rrbracket^{\text{L}}_{\rrbracket})
\end{array}$$

does not exhibit inputs or outputs, so, in particular, $(\nu\vec{\alpha}\vec{y}\vec{c}c')\,(\llbracket K \rrbracket^{\text{L}}_{\rrbracket} c'\,\{c_1/\gamma\})$ (the only one sub-process that could) does not, and thereby neither does $(\nu\vec{\alpha}\vec{y}\vec{c}c')\,(\llbracket K \rrbracket^{\text{L}}_{\rrbracket} c')$, so neither does $(\nu\vec{\alpha}\vec{y}\vec{c}c')\,(\llbracket K \rrbracket^{\text{L}}_{\rrbracket} \alpha)$ so also

$$\begin{array}{l}
(\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,(\llbracket K \rrbracket^{\text{L}}_{\rrbracket} \alpha \mid \llbracket \alpha := P_1 \cdot c_2 \rrbracket^{\text{L}}_{\rrbracket} \mid \overrightarrow{\llbracket c_i := P_i \cdot c_{i+1} \rrbracket^{\text{L}}_{\rrbracket}}) \mid \llbracket \boldsymbol{S} \rrbracket^{\text{L}}_{\rrbracket}) \\
\quad = \;\; (\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,(\llbracket K \rrbracket^{\text{L}}_{\rrbracket} \alpha\,\{c_1/\alpha\} \mid \llbracket c_1 := P_1 \cdot c_2 \rrbracket^{\text{L}}_{\rrbracket} \mid \overrightarrow{\llbracket c_i := P_i \cdot c_{i+1} \rrbracket^{\text{L}}_{\rrbracket}}) \mid \llbracket \boldsymbol{S} \rrbracket^{\text{L}}_{\rrbracket}) \\
\quad \approx_{\text{D}} \;\; (\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,(\llbracket K \rrbracket^{\text{L}}_{\rrbracket} \alpha\,\{c_1/\alpha\} \mid \overrightarrow{\llbracket c_i := P_i \cdot c_{i+1} \rrbracket^{\text{L}}_{\rrbracket}}) \mid \llbracket \boldsymbol{S} \rrbracket^{\text{L}}_{\rrbracket}) \\
\quad \triangleq \;\; \llbracket (\mu\alpha.[\alpha \rrbracket^{\text{L}}_{\rrbracket} K)P_1 \boldsymbol{S}_2 P_2 \boldsymbol{S}_1 \cdots P_n \boldsymbol{S}_n]a
\end{array}$$

does not.

$$
\begin{array}{rl}
\llbracket \Delta\Delta_{\rrbracket}^{\mathrm{L}} a \;=\; & \llbracket (\lambda x.xx)\Delta_{\rrbracket}^{\mathrm{L}} a \;\triangleq\; (\nu c)\,((\nu xb)\,(\llbracket xx_{\rrbracket}^{\mathrm{L}} b \mid \overline{c}\langle x,b\rangle) \mid !c(v,d).(\llbracket v{:=}\Delta_{\rrbracket}^{\mathrm{L}} \mid !d{\to}\overline{a})) \\
\to(c) & (\nu xb)\,(\llbracket xx_{\rrbracket}^{\mathrm{L}} b \mid \llbracket x{:=}\Delta_{\rrbracket}^{\mathrm{L}} \mid !b{\to}\overline{a}) \mid (\nu c)\,(!!c(v,d).(\llbracket v{:=}\Delta_{\rrbracket}^{\mathrm{L}} \mid !d{\to}\overline{a})) \\
\triangleq,\approx_{\mathrm{G}} & (\nu xb)\,((\nu c)\,(x(u).!u{\to}\overline{c} \mid !c(v,d).(\llbracket v{:=}x_{\rrbracket}^{\mathrm{L}} \mid !d{\to}\overline{b})) \mid !\overline{x}(w).\llbracket \Delta_{\rrbracket}^{\mathrm{L}} w \mid !b{\to}\overline{a}) \\
\to(x) & (\nu xbw)\,((\nu c)\,(!w{\to}\overline{c} \mid !c(v,d).(\llbracket v{:=}x_{\rrbracket}^{\mathrm{L}} \mid !d{\to}\overline{b})) \mid (\nu yb)\,(\llbracket yy_{\rrbracket}^{\mathrm{L}} b \mid \overline{w}\langle y,b\rangle) \mid \llbracket x{:=}\Delta_{\rrbracket}^{\mathrm{L}} \mid !b{\to}\overline{a}) \\
\to(w) & (\nu xb)\,((\nu c)\,((\nu yb)\,(\llbracket yy_{\rrbracket}^{\mathrm{L}} b \mid \overline{c}\langle y,b\rangle) \mid !c(v,d).(\llbracket v{:=}x_{\rrbracket}^{\mathrm{L}} \mid !d{\to}\overline{b}) \mid \llbracket x{:=}\Delta_{\rrbracket}^{\mathrm{L}} \mid !b{\to}\overline{a})) \\
\to(c),\approx_{\mathrm{G}} & (\nu xb)\,((\nu yb_1)\,(\llbracket yy_{\rrbracket}^{\mathrm{L}} b_1 \mid \llbracket y{:=}x_{\rrbracket}^{\mathrm{L}} \mid !b_1{\to}\overline{b}) \mid \llbracket x{:=}\Delta_{\rrbracket}^{\mathrm{L}} \mid !b{\to}\overline{a}) \\
\triangleq & (\nu xb)\,((\nu yb_1)\,((\nu c)\,(y(u).!u{\to}\overline{c} \mid !c(v,d).(\llbracket v{:=}y_{\rrbracket}^{\mathrm{L}} \mid !d{\to}\overline{b_1})) \mid \\
& \qquad\qquad !\overline{y}(w).\llbracket x_{\rrbracket}^{\mathrm{L}} w \mid !b_1{\to}\overline{b}) \mid \llbracket x{:=}\Delta_{\rrbracket}^{\mathrm{L}} \mid !b{\to}\overline{a}) \\
\to(y) & (\nu xb)\,((\nu yb_1)\,((\nu c)\,(w{\to}\overline{c} \mid !c(v,d).(\llbracket v{:=}y_{\rrbracket}^{\mathrm{L}} \mid !d{\to}\overline{b_1})) \mid \llbracket x_{\rrbracket}^{\mathrm{L}} w \mid \\
& \qquad\qquad \llbracket y{:=}x_{\rrbracket}^{\mathrm{L}} \mid !b_1{\to}\overline{b}) \mid \llbracket x{:=}\Delta_{\rrbracket}^{\mathrm{L}} \mid !b{\to}\overline{a}) \\
\triangleq & (\nu xb)\,((\nu yb_1)\,((\nu c)\,(w{\to}\overline{c} \mid !c(v,d).(\llbracket v{:=}y_{\rrbracket}^{\mathrm{L}} \mid !d{\to}\overline{b_1})) \mid \\
& \qquad\qquad x(u).!u{\to}\overline{w} \mid \llbracket y{:=}x_{\rrbracket}^{\mathrm{L}} \mid !b_1{\to}\overline{b}) \mid \llbracket x{:=}\Delta_{\rrbracket}^{\mathrm{L}} \mid !b{\to}\overline{a}) \\
\equiv & (\nu xb)\,((\nu yb_1)\,((\nu c)\,(w{\to}\overline{c} \mid !c(v,d).(\llbracket v{:=}y_{\rrbracket}^{\mathrm{L}} \mid !d{\to}\overline{b_1})) \mid x(u).!u{\to}\overline{w} \mid \\
& \qquad\qquad \llbracket y{:=}x_{\rrbracket}^{\mathrm{L}} \mid !b_1{\to}\overline{b}) \mid !\overline{x}(w).(\nu zb)\,(\llbracket zz_{\rrbracket}^{\mathrm{L}} b \mid \overline{w}\langle z,b\rangle) \mid \llbracket x{:=}\Delta_{\rrbracket}^{\mathrm{L}} \mid !b{\to}\overline{a}) \\
\to(x,w_1,w,c) & (\nu xb)\,((\nu yb_1)\,((\nu zb_2)\,(\llbracket zz_{\rrbracket}^{\mathrm{L}} b_2 \mid \llbracket z{:=}y_{\rrbracket}^{\mathrm{L}} \mid !b_2{\to}\overline{b_1})) \mid \llbracket y{:=}x_{\rrbracket}^{\mathrm{L}} \mid !b_1{\to}\overline{b} \mid \llbracket x{:=}\Delta_{\rrbracket}^{\mathrm{L}} \mid !b{\to}\overline{a}) \\
\cdots &
\end{array}
$$

Figure 5: Running $\llbracket \Delta\Delta_{\rrbracket}^{\mathrm{L}} a$ without renaming, but using garbage collection.

(μ_r): Then $R = \mu\alpha.[\beta]K$, and M contracts to $(\mu\gamma.[\beta]K\langle\alpha{:=}P_1{\cdot}\gamma\rangle)\boldsymbol{S}_2P_2\boldsymbol{S}_1\cdots P_n\boldsymbol{S}_n$. Since, as in the proof of Thm 7.1,

$$
\begin{array}{rl}
\llbracket (\mu\alpha.[\beta]K)P_1\boldsymbol{S}_2P_2\boldsymbol{S}_1\cdots P_n\boldsymbol{S}_n{}_{\rrbracket}^{\mathrm{L}} a & \\
\triangleq,\equiv & (\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,((\nu c_1)\,(\llbracket \mu\alpha.[\beta]K_{\rrbracket}^{\mathrm{L}} c_1 \mid \llbracket c_1{:=}P_1{\cdot}c_2{}_{\rrbracket}^{\mathrm{L}}) \mid \overrightarrow{\llbracket c_i{:=}P_i{\cdot}c_{i+1}{}_{\rrbracket}^{\mathrm{L}}}) \mid \llbracket \boldsymbol{S}_{\rrbracket}^{\mathrm{L}}) \\
\triangleq & (\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,(\llbracket K_{\rrbracket}^{\mathrm{L}}\beta\{c_1/\alpha\} \mid \llbracket c_1{:=}P_1{\cdot}c_2{}_{\rrbracket}^{\mathrm{L}} \mid \overrightarrow{\llbracket c_i{:=}P_i{\cdot}c_{i+1}{}_{\rrbracket}^{\mathrm{L}}}) \mid \llbracket \boldsymbol{S}_{\rrbracket}^{\mathrm{L}}) \\
=_\alpha & (\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,(\llbracket K_{\rrbracket}^{\mathrm{L}}\beta \mid \llbracket \alpha{:=}P_1{\cdot}c_2{}_{\rrbracket}^{\mathrm{L}} \mid \overrightarrow{\llbracket c_i{:=}P_i{\cdot}c_{i+1}{}_{\rrbracket}^{\mathrm{L}}}) \mid \llbracket \boldsymbol{S}_{\rrbracket}^{\mathrm{L}}) \\
\triangleq & (\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,((\nu\alpha)\,(\llbracket K_{\rrbracket}^{\mathrm{L}}\beta \mid \llbracket \alpha{:=}P_1{\cdot}\gamma_{\rrbracket}^{\mathrm{L}})\{c_2/\gamma\} \mid \overrightarrow{\llbracket c_i{:=}P_i{\cdot}c_{i+1}{}_{\rrbracket}^{\mathrm{L}}}) \mid \llbracket \boldsymbol{S}_{\rrbracket}^{\mathrm{L}}) \\
\triangleq,\equiv & \llbracket (\mu\gamma.[\beta]K\langle\alpha{:=}P_1{\cdot}\gamma\rangle)\boldsymbol{S}_2P_2\boldsymbol{S}_1\cdots P_n\boldsymbol{S}_n{}_{\rrbracket}^{\mathrm{L}} a
\end{array}
$$

the result follows immediately by co-induction.

(hv): Then $R = x$ and $\langle x{:=}N\rangle \in \boldsymbol{S}$, and M contracts to $N\boldsymbol{S}_0P_1\boldsymbol{S}_1\cdots P_n\boldsymbol{S}_n$. By coinduction, the process

$$\llbracket N\boldsymbol{S}_0P_1\boldsymbol{S}_1\cdots P_n\boldsymbol{S}_n{}_{\rrbracket}^{\mathrm{L}} a \;\triangleq\; (\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,(\llbracket N_{\rrbracket}^{\mathrm{L}} c_1 \mid \overrightarrow{\llbracket c_i{:=}P_i{\cdot}c_{i+1}{}_{\rrbracket}^{\mathrm{L}}}) \mid \llbracket \boldsymbol{S}_{\rrbracket}^{\mathrm{L}})$$

does not exhibit inputs or outputs, so, in particular, the process

$$\llbracket x\boldsymbol{S}_1P_1\boldsymbol{S}_2P_2\boldsymbol{S}_1\cdots P_n\boldsymbol{S}_n{}_{\rrbracket}^{\mathrm{L}} a \;\triangleq,\equiv\; (\nu\vec{\alpha}\vec{y})\,((\nu\vec{c})\,(x(u).!u{\to}\overline{c_1} \mid \overrightarrow{\llbracket c_i{:=}P_i{\cdot}c_{i+1}{}_{\rrbracket}^{\mathrm{L}}}) \mid \overline{x}(w).\llbracket N_{\rrbracket}^{\mathrm{L}} w \mid \llbracket \boldsymbol{S}_{\rrbracket}^{\mathrm{L}})$$

where $x \in \vec{y}$, does not. □

The reduction without renaming of $\llbracket \Delta\Delta_{\rrbracket}^{\mathrm{L}} a$ is given in Fig. 5, and shows that the interpretation of $\Delta\Delta$ reduces without creating output over a.

As a direct consequence of Lemma 8.7, as for Milner's and Sangiorgi's interpretations, our interpretation is not extensional, since $\llbracket \Delta\Delta_{\rrbracket}^{\mathrm{L}} a \approx 0$, whereas

$$\llbracket \lambda x.\Delta\Delta x_{\rrbracket}^{\mathrm{L}} a \;\triangleq\; (\nu xb)\,(\llbracket \Delta\Delta x_{\rrbracket}^{\mathrm{L}} b \mid \overline{a}\langle x,b\rangle) \;\not\approx\; 0.$$

We can show that if a term reduces to an abstraction (perhaps with a preceding context switch), then its interpretation creates an output, and that if it runs to a term with a head variable, its interpretation creates an input.

Lemma 8.8 *i) If $M \to_{w\mathrm{XH}}^{nf} \lambda x.N$, then $\llbracket M_{\rrbracket}^{\mathrm{L}} a \Downarrow \overline{a}$.*

ii) If $M \to_{w\mathrm{XH}}^{nf} \mu\alpha.[\beta]\lambda x.N$, then $\llbracket M_{\rrbracket}^{\mathrm{L}} a \Downarrow \overline{\beta}$.

iii) If $M \to_{w\mathrm{XH}}^{nf} xN_1\boldsymbol{S}_1\cdots N_n\boldsymbol{S}_n$ or $M \to_{w\mathrm{XH}}^{nf} \mu\alpha.[\beta]xN_1\boldsymbol{S}_1\cdots N_n\boldsymbol{S}_1$, then $\llbracket M_{\rrbracket}^{\mathrm{L}} a \Downarrow x$.

Proof: Since '$\to_{w\mathrm{XH}}^{*}$' $\subseteq$ '$\to_{\mathrm{XH}}^{*}$', this follows from Thm 7.1, and the observation that the resulting processes (obtained by encoding the normal forms) do indeed exhibit the input or output. □

As to the reverse, we can show that if the interpretation of M produces an output, then M reduces by head reduction to an abstraction; similarly, if the interpretation of M produces an input, then M reduces by head reduction to a term with a head variable.

Lemma 8.9 *i) If* $\llbracket M \rrbracket^{\mathrm{L}}_{\mathbb{J}} a \Downarrow \overline{a}$*, then there exist* x, $N \in \lambda\mu\mathbf{x}$ *such that* $\llbracket M \rrbracket^{\mathrm{L}}_{\mathbb{J}} a \approx \llbracket \lambda x.N \rrbracket^{\mathrm{L}}_{\mathbb{J}} a$*, and* $M \to^{nf}_{w\mathrm{XH}} \lambda x.N$*.*

ii) If $\llbracket M \rrbracket^{\mathrm{L}}_{\mathbb{J}} a \Downarrow \overline{c}$*, with* $a \neq c$*, then there exist* $\alpha, x, N \in \lambda\mu\mathbf{x}$ *such that* $\llbracket M \rrbracket^{\mathrm{L}}_{\mathbb{J}} a \approx \llbracket \mu\alpha.[c]\lambda x.N \rrbracket^{\mathrm{L}}_{\mathbb{J}} a \triangleq \llbracket \lambda x.N \rrbracket^{\mathrm{L}}_{\mathbb{J}} c\,\{a/\alpha\}$*, and* $M \to^{nf}_{w\mathrm{XH}} \mu\alpha.[c]\lambda x.N$*.*

iii) If $\llbracket M \rrbracket^{\mathrm{L}}_{\mathbb{J}} a \not\Downarrow_o$ *but* $\llbracket M \rrbracket^{\mathrm{L}}_{\mathbb{J}} a \Downarrow x$*, then there exist* $N_1, \ldots, N_n$*,* c*, and* $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_n$ *with* $n \geq 0$ *such that:*

- $\llbracket M \rrbracket^{\mathrm{L}}_{\mathbb{J}} a \approx \llbracket x\boldsymbol{S}_0 N_1 \boldsymbol{S}_1 \cdots N_n \boldsymbol{S}_n \rrbracket^{\mathrm{L}}_{\mathbb{J}} c$*;*
- $M \to^{nf}_{w\mathrm{XH}} x\boldsymbol{S}_0 N_1 \boldsymbol{S}_1 \cdots N_n \boldsymbol{S}_n$ *if* $a = c$*;*
- $M \to^{nf}_{w\mathrm{XH}} \mu\alpha.[c]x\boldsymbol{S}_0 N_1 \boldsymbol{S}_1 \cdots N_n \boldsymbol{S}_n$*, if* $a \neq c$*.*

Proof: *i*) By checking the proof for Thm 7.5, we observe that if $\llbracket M \rrbracket^{\mathrm{L}}_{\mathbb{J}} a$ exhibits an output, then, using explicit head reduction, M reduces to an abstraction. But then M also runs to an abstraction using weak explicit head reduction, so there exist x, N such that $M \to^{nf}_{w\mathrm{XH}} \lambda x.N$. Since '$\to_{w\mathrm{XH}}$' $\subseteq$ '$\to_{\mathrm{XH}}$', also $M \to^{*}_{\mathrm{XH}} \lambda x.N$, and by Thm 7.1 we get $\llbracket M \rrbracket^{\mathrm{L}}_{\mathbb{J}} a \approx \llbracket \lambda x.N \rrbracket^{\mathrm{L}}_{\mathbb{J}} a$.

ii) As in the previous case; the output name can only change because of a context switch.

iii) If $\llbracket M \rrbracket^{\mathrm{L}}_{\mathbb{J}} a$ runs to a process that inputs, but does not output, then by the proof for Thm 7.5 and Remark 7.4 M runs to a term with a head variable and without outermost abstractions. But then there also exist $N_1, \ldots, N_n$, c, and $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_n$ with $n \geq 0$ such that $M \to^{nf}_{w\mathrm{XH}} x\boldsymbol{S}_0 N_1 \boldsymbol{S}_1 \cdots N_n \boldsymbol{S}_n$. Since also $M \to^{*}_{\mathrm{XH}} x\boldsymbol{S}_0 N_1 \boldsymbol{S}_1 \cdots N_n \boldsymbol{S}_n$, we get $\llbracket M \rrbracket^{\mathrm{L}}_{\mathbb{J}} a \approx \llbracket x\boldsymbol{S}_0 N_1 \boldsymbol{S}_1 \cdots N_n \boldsymbol{S}_n \rrbracket^{\mathrm{L}}_{\mathbb{J}} a$ by Thm 7.1. The case $a \neq c$ is similar. □

9 On renaming

By Thm 7.1, renaming might be used during the simulation of $\lambda\mu$-reduction. However, in this section we will show that we can do without renaming when simulating lazy reductions for closed terms, thereby emulating Milner's result (Thm 3.3). As a consequence, it is safe to say that renaming is the price we pay for the capability to deal with reductions under abstraction, and thereby that of open terms. As an illustration of this fact, notice that, as shown in Fig. 6, we *can* run the π-process $\llbracket (\lambda x.xx)\,(\lambda y.y) \rrbracket^{\mathrm{L}}_{\mathbb{J}} a$ without using renaming; there we perform the two substitutions without resorting to the renaming of outputs of translated λ-terms. Notice that we could also have postponed all '$\approx_{\mathrm{G}}$' steps until the end.

Example 9.1 When modelling head reduction, we cannot do without renaming completely, not even for closed terms:

$$\begin{array}{rcl}
\llbracket \lambda x.(\lambda y.y)\,x \rrbracket^{\mathrm{L}}_{\mathbb{J}} a & \triangleq & (\nu xb)\,((\nu c)\,((\nu yb')\,(\llbracket y \rrbracket^{\mathrm{L}}_{\mathbb{J}} b' \mid \overline{c}\langle y,b'\rangle) \mid !c(v,d).(\llbracket v := x \rrbracket^{\mathrm{L}}_{\mathbb{J}} \mid !d \to \overline{b})) \mid \overline{a}\langle x,b\rangle) \\
& \to_{\pi},\approx_{\mathrm{G}} & (\nu xb)\,((\nu yb')\,(\llbracket y \rrbracket^{\mathrm{L}}_{\mathbb{J}} b' \mid \llbracket y := x \rrbracket^{\mathrm{L}}_{\mathbb{J}} \mid !b' \to \overline{b}) \mid \overline{a}\langle x,b\rangle) \\
& \triangleq & (\nu xb)\,((\nu yb')\,(y(u).!u \to \overline{b'} \mid !\overline{y}(w).\llbracket x \rrbracket^{\mathrm{L}}_{\mathbb{J}} w \mid !b' \to \overline{b}) \mid \overline{a}\langle x,b\rangle) \\
& \to_{\pi},\approx_{\mathrm{G}} & (\nu xb)\,((\nu b')\,((\nu w)\,(!w \to \overline{b'} \mid \llbracket x \rrbracket^{\mathrm{L}}_{\mathbb{J}} w) \mid !b' \to \overline{b}) \mid \overline{a}\langle x,b\rangle) \\
& \triangleq & (\nu xb)\,((\nu b')\,((\nu w)\,(!w \to \overline{b'} \mid x(u).!u \to \overline{w}) \mid !b' \to \overline{b}) \mid \overline{a}\langle x,b\rangle)
\end{array}$$

We would like this to reduce to $(\nu xb)\,(x(u).!u \to \overline{b} \mid \overline{a}\langle x,b\rangle) \triangleq \llbracket \lambda x.x \rrbracket^{\mathrm{L}}_{\mathbb{J}} a$, but it cannot; the last process above is irreducible. We would therefore need renaming to achieve

$$\begin{array}{rll}
& (\nu xb)\,((\nu b')\,((\nu w)\,(!w \to \overline{b'} \mid x(u).!u \to \overline{w}) \mid !b' \to \overline{b}) \mid \overline{a}\langle x,b\rangle) & \\
\approx_{\mathrm{R}} & (\nu xb)\,((\nu b')\,(x(u).!u \to \overline{b'} \mid !b' \to \overline{b}) \mid \overline{a}\langle x,b\rangle) & \\
\approx_{\mathrm{R}} & (\nu xb)\,(x(u).!u \to \overline{b} \mid \overline{a}\langle x,b\rangle) & \triangleq \quad \llbracket \lambda x.x \rrbracket^{\mathrm{L}}_{\mathbb{J}} a
\end{array}$$

However, we can show that we do not need renaming when interpreting a *weak reduction to normal form on closed terms*, as defined in Definition 8.2.

$$
\begin{array}{rl}
& \llbracket (\lambda x.xx)\,(\lambda y.y)\rrbracket^{\mathsf{L}}\, a \\
\triangleq & (\nu c)\,((\nu xb)\,(\llbracket xx\rrbracket^{\mathsf{L}}\, b \mid \overline{c}\langle x,b\rangle) \mid !\,c(v,d).(\llbracket v{:=}\lambda y.y\rrbracket^{\mathsf{L}} \mid !\,d{\to}\overline{a})) \\
\to_\pi (c) & (\nu x)\,(\llbracket xx\rrbracket^{\mathsf{L}}\, b \mid \llbracket x{:=}\lambda y.y\rrbracket^{\mathsf{L}} \mid b{\to}\overline{a} \mid (\nu c)\,(!\,c(v,d).(\llbracket v{:=}\lambda y.y\rrbracket^{\mathsf{L}} \mid d{\to}\overline{b}))) \\
\approx_{\mathrm{G}} & (\nu x)\,(\llbracket xx\rrbracket^{\mathsf{L}}\, b \mid \llbracket x{:=}\lambda y.y\rrbracket^{\mathsf{L}} \mid b{\to}\overline{a}) \\
\triangleq & (\nu x)\,((\nu c)\,(x(u).!\,u{\to}\overline{c} \mid !\,c(v,d).(\llbracket v{:=}x\rrbracket^{\mathsf{L}} \mid !\,d{\to}\overline{b})) \mid !\,\overline{x}(w).\llbracket\lambda y.y\rrbracket^{\mathsf{L}}\, w \mid b{\to}\overline{a}) \\
\to_\pi (x), \triangleq & (\nu xb)\,((\nu c)\,((\nu w)\,((\nu yb_1)\,(\llbracket y\rrbracket^{\mathsf{L}}\, b_1 \mid \overline{w}\langle y,b_1\rangle) \mid !\,w{\to}\overline{c}) \mid !\,c(v,d).(\llbracket v{:=}x\rrbracket^{\mathsf{L}} \mid d{\to}\overline{b})) \mid \\
& \qquad\qquad \llbracket x{:=}\lambda y.y\rrbracket^{\mathsf{L}} \mid b{\to}\overline{a}) \\
\to_\pi (w), \approx_{\mathrm{G}} & (\nu xb)\,((\nu c)\,((\nu yb_1)\,(\llbracket y\rrbracket^{\mathsf{L}}\, b_1 \mid \overline{c}\langle y,b_1\rangle) \mid !\,c(v,d).(\llbracket v{:=}x\rrbracket^{\mathsf{L}} \mid d{\to}\overline{b})) \mid \llbracket x{:=}\lambda y.y\rrbracket^{\mathsf{L}} \mid b{\to}\overline{a}) \\
\to_\pi (c), \triangleq, \approx_{\mathrm{G}} & (\nu x)\,((\nu b_1)\,(y(u).!\,u{\to}\overline{b_1} \mid !\,y(w).\llbracket x\rrbracket^{\mathsf{L}}\, w \mid b_1{\to}\overline{a}) \mid !\,\overline{x}(w).\llbracket\lambda y.y\rrbracket^{\mathsf{L}}\, w) \\
\to_\pi (y) & (\nu xb)\,((\nu b_1)\,((\nu w)\,(w{\to}\overline{b_1} \mid \llbracket x\rrbracket^{\mathsf{L}}\, w) \mid (\nu y)\,(\llbracket y{:=}x\rrbracket^{\mathsf{L}}) \mid b_1{\to}\overline{b}) \mid \llbracket x{:=}\lambda y.y\rrbracket^{\mathsf{L}} \mid b{\to}\overline{a}) \\
\triangleq, \approx_{\mathrm{G}} & (\nu xb)\,((\nu b_1)\,((\nu w)\,(w{\to}\overline{b_1} \mid x(u).!\,u{\to}\overline{w}) \mid b_1{\to}\overline{b}) \mid \\
& \qquad\qquad \overline{x}(w_1).\llbracket\lambda y.y\rrbracket^{\mathsf{H}}\, w_1 \mid !\,\overline{x}(w).\llbracket\lambda y.y\rrbracket^{\mathsf{L}}\, w \mid b{\to}\overline{a}) \\
\to_\pi (x), \approx_{\mathrm{G}} & (\nu b)\,((\nu w_1)\,((\nu b_1)\,((\nu w)\,(w{\to}\overline{b_1} \mid w_1{\to}\overline{w}) \mid b_1{\to}\overline{b}) \mid \llbracket\lambda y.y\rrbracket^{\mathsf{L}}\, w_1) \mid b{\to}\overline{a}) \\
\triangleq & (\nu b)\,((\nu w_1)\,((\nu yb_1)\,((\nu w)\,(w{\to}\overline{b_1} \mid w_1{\to}\overline{w}) \mid b_1{\to}\overline{b}) \mid (\nu yb_2)\,(\llbracket y\rrbracket^{\mathsf{L}}\, b_2 \mid \overline{w_1}\langle y,b_2\rangle))) \mid b{\to}\overline{a}) \\
\to_\pi (w_1 w b_1 b), \triangleq & (\nu yb_2)\,(\llbracket y\rrbracket^{\mathsf{L}}\, b_2 \mid \overline{a}\langle y,b_2\rangle) \;\triangleq\; \llbracket\lambda y.y\rrbracket^{\mathsf{L}}\, a
\end{array}
$$

Figure 6: Running $\llbracket (\lambda x.xx)\,(\lambda y.y)\rrbracket^{\mathsf{L}}\, a \to_\pi^* \llbracket\lambda y.y\rrbracket^{\mathsf{L}}\, a$ without renaming.

Theorem 9.2 ($\llbracket\cdot\rrbracket^{\mathsf{L}}\cdot$ PRESERVES '$\to_{w\mathrm{XH}}$' WITHOUT RENAMING) *If P is a closed $\lambda\mu$**x**-term, and $P \to_{w\mathrm{XH}}^{nf} Q$ (so either $Q = \lambda z.Q'$ or $Q = \mu\alpha.[\alpha]\lambda z.Q'$, with $\alpha \in Q$), then there exists P such that $\llbracket P\rrbracket^{\mathsf{L}}\, a \to_\pi^{nf} \mathsf{P}$ and $\mathsf{P} \approx_{\mathrm{G}}, \approx_{\mathrm{D}} \llbracket Q\rrbracket^{\mathsf{L}}\, a$.*

Proof: We follow the structure of the proof of Thm 7.1, where we focus on the cases that use renaming. Since $\llbracket\mu\alpha.[\alpha]P'\rrbracket^{\mathsf{L}}\, a \triangleq \llbracket P'\rrbracket^{\mathsf{L}}\, \alpha\,\{a/\alpha\} = \llbracket P'\,\{a/\alpha\}\rrbracket^{\mathsf{L}}\, \alpha$, we can assume that P does not start with a context switch.

$P = (\lambda x.M)\,N\,\boldsymbol{S}_1 M_2 \boldsymbol{S}_2 \cdots M_n \boldsymbol{S}_n$: Let $\boldsymbol{S} = \boldsymbol{S}_1 \ldots \boldsymbol{S}_n$.

$$
\begin{array}{rl}
& \llbracket (\lambda x.M)\,N\,\boldsymbol{S}_1 M_2 \boldsymbol{S}_2 \cdots M_n \boldsymbol{S}_n\rrbracket^{\mathsf{L}}\, c_{n+1} \\
\triangleq, \equiv (6.6) & (\nu\vec{y}\vec{\alpha}\vec{c})\,(\llbracket\lambda x.M\rrbracket^{\mathsf{L}}\, c_1 \mid \overrightarrow{\llbracket c_i{:=}M_i{\cdot}c_{i+1}\rrbracket^{\mathsf{L}}} \mid \llbracket\boldsymbol{S}\rrbracket^{\mathsf{L}}) \\
\equiv & (\nu\vec{y}\vec{\alpha}\vec{c})\,((\nu bx)\,(\llbracket M\rrbracket^{\mathsf{L}}\, b \mid \overline{c_1}\langle x,b\rangle) \mid !\,c_1(v,d).(\llbracket v{:=}N\rrbracket^{\mathsf{L}} \mid !\,d{\to}\overline{c_2}) \mid \overrightarrow{\llbracket c_i{:=}M_i{\cdot}c_{i+1}\rrbracket^{\mathsf{L}}} \mid \llbracket\boldsymbol{S}\rrbracket^{\mathsf{L}}) \\
\to_\pi (c_1) & (\nu\vec{y}\vec{\alpha}\vec{c})\,((\nu bx)\,(\llbracket M\rrbracket^{\mathsf{L}}\, b \mid \llbracket x{:=}N\rrbracket^{\mathsf{L}} \mid !\,b{\to}\overline{c_2}) \mid (\nu c_1)\,(\llbracket c_1{:=}N{\cdot}c_2\rrbracket^{\mathsf{L}}) \mid \overrightarrow{\llbracket c_i{:=}M_i{\cdot}c_{i+1}\rrbracket^{\mathsf{L}}} \mid \llbracket\boldsymbol{S}\rrbracket^{\mathsf{L}})
\end{array}
$$

At this point the proof of Thm 7.1 uses renaming to obtain $\llbracket M\rrbracket^{\mathsf{L}}\, e$, since it might be that M evaluates to a variable y. All synchronisations in the later process take place inside $\llbracket M\rrbracket^{\mathsf{L}}\, b$, until that process outputs (on b). By Thm 7.5, we know that $\llbracket M\rrbracket^{\mathsf{L}}\, b$ will run to P such that $\mathsf{P} \approx_{\mathrm{RGD}} \llbracket N\rrbracket^{\mathsf{L}}\, b$ with $M \to_{\mathrm{XH}}^{nf} N$.

Since $P \to_{w\mathrm{XH}}^{nf} Q$, P has a WXHNF, the $\to_{w\mathrm{XH}}$-reduction on M terminates as well, and N is either an abstraction, an applicative term starting with a variable, or either one of those preceded by a context switch.

In case M evaluates to an abstraction $\lambda y.M'$, or $\mu\alpha.[\alpha]\lambda y.M'$ we get

$$
\begin{array}{rl}
& (\nu bx)\,(\llbracket M\rrbracket^{\mathsf{L}}\, b \mid !\,b{\to}\overline{e} \mid \llbracket x{:=}N\rrbracket^{\mathsf{L}}) \\
\to_\pi^* & (\nu bx)\,(\llbracket\lambda y.M'\rrbracket^{\mathsf{L}}\, b \mid \mathsf{G} \mid !\,b{\to}\overline{e} \mid \llbracket x{:=}N\rrbracket^{\mathsf{L}}) \\
\triangleq & (\nu bx)\,((\nu yb')\,(\llbracket M'\rrbracket^{\mathsf{L}}\, b' \mid \overline{b}\langle y,b'\rangle) \mid \mathsf{G} \mid !\,b{\to}\overline{e} \mid \llbracket x{:=}N\rrbracket^{\mathsf{L}}) \\
\to_\pi (b) & (\nu x)\,((\nu yb')\,(\llbracket M'\rrbracket^{\mathsf{L}}\, b' \mid \overline{e}\langle y,b'\rangle) \mid \llbracket x{:=}N\rrbracket^{\mathsf{L}}) \mid (\nu b)\,(!\,b{\to}\overline{e}) \mid \mathsf{G}
\end{array}
$$

where G is garbage, so the renaming is not necessary, and garbage collection can be delayed. In case N starts with a variable, we find ourselves in the second case.

$P = x\boldsymbol{S}_1 M_1 \cdots \boldsymbol{S}_n M_n \boldsymbol{S}_{n+1}$, *with $\langle x{:=}N\rangle \in \boldsymbol{S}_k$, for some $1 \le k \le n+1$:*

$$
\begin{array}{rll}
\llbracket x\boldsymbol{S}_1 M_1 \cdots \boldsymbol{S}_n M_n \boldsymbol{S}_{n+1}\rrbracket^{\mathsf{L}}\, c_{n+1} & \equiv & (6.6) \\
(\nu\vec{y}\vec{\alpha})\,(\llbracket x\rrbracket^{\mathsf{L}}\, c_1 \mid \overrightarrow{\llbracket c_i{:=}M_i{\cdot}c_{i+1}\rrbracket^{\mathsf{L}}} \mid \llbracket\boldsymbol{S}\rrbracket^{\mathsf{L}}) & \triangleq, \equiv & (!\,\overline{x}(w).\llbracket N\rrbracket^{\mathsf{L}}\, w \in \llbracket\boldsymbol{S}\rrbracket^{\mathsf{L}}) \\
(\nu\vec{y}\vec{\alpha})\,(x(u).!\,u{\to}\overline{c_1} \mid \overrightarrow{\llbracket c_i{:=}M_i{\cdot}c_{i+1}\rrbracket^{\mathsf{L}}} \mid \overline{x}(w).\llbracket N\rrbracket^{\mathsf{L}}\, w \mid \llbracket\boldsymbol{S}\rrbracket^{\mathsf{L}}) & \to_\pi & (x) \\
(\nu\vec{y}\vec{\alpha})\,((\nu w)\,(\llbracket N\rrbracket^{\mathsf{L}}\, w \mid !\,w{\to}\overline{c_1}) \mid \overrightarrow{\llbracket c_i{:=}M_i{\cdot}c_{i+1}\rrbracket^{\mathsf{L}}} \mid \llbracket\boldsymbol{S}\rrbracket^{\mathsf{L}}) & &
\end{array}
$$

Also at this point the proof of Thm 7.1 uses renaming. In case N evaluates to an abstraction we get a situation similar to the previous case. Otherwise, it reduces to (a term starting with) a variable bound by a substitution, which is dealt with in this case. $\square$

So weak explicit head reduction on a closed $\lambda\mu\mathbf{x}$-term P either generates an (interpretation of an) abstraction before a forwarder, or a (term starting with a) variable before a forwarder, that eventually will be replaced by an abstraction.

So, when simulating weak explicit reduction to normal form on closed terms, renaming can be postponed, and the relation '$\approx_{\mathrm{R}}$' is not needed.

10 Weak equivalences for $\lambda\mu$ and $\lambda\mu\mathbf{x}$

We will now define notions of weak equivalences '$\sim_{w\beta\mu}$' and '$\sim_{w\mathrm{H}}$' between terms of $\lambda\mu$, and '$\sim_{w\mathrm{xH}}$' between terms of $\lambda\mu\mathbf{x}$ (the last two are defined coinductively as bisimulations) that are based on weak reduction, and show that the last two equate the same pure $\lambda\mu$-terms. These notions all consider terms without WHNF equivalent.

First we define a weak equivalence generated by the reduction relation '$\rightarrow_{w\beta\mu}$'.

Definition 10.1 We define '$\sim_{w\beta\mu}$' as the smallest congruence that contains:

$$\begin{array}{rcll} M, N \textit{ have no } \textsc{whnf} & \Rightarrow & M \sim_{w\beta\mu} N & \\ (\lambda x.M)\,N & \sim_{w\beta\mu} & M\{N/x\} & \\ (\mu\alpha.\mathrm{C})N & \sim_{w\beta\mu} & \mu\gamma.\mathrm{C}\{N\cdot\gamma/\alpha\} & (\gamma \textit{ fresh}) \\ \mu\alpha.[\beta]\mu\gamma.[\delta]M & \sim_{w\beta\mu} & \mu\alpha.[\delta]M\{\beta/\gamma\} & \\ \mu\alpha.[\alpha]M & \sim_{w\beta\mu} & M & (\alpha \notin M) \end{array}$$

Notice that $\Delta\Delta \sim_{w\beta\mu} \Omega\Omega$ since both have no WHNF, and $\lambda z.\Delta\Delta \sim_{w\beta\mu} \lambda z.\Omega\Omega$ since both are abstractions over terms that are in '$\sim_{w\beta\mu}$' and that relation is a congruence, but $\Delta\Delta \neq_{\beta\mu} \Omega\Omega$; moreover, '$\sim_{w\beta\mu}$' is closed under reduction.

Since reduction is confluent, the following is immediate.

Proposition 10.2 If $M \sim_{w\beta\mu} N$ and $M \rightarrow^*_{w\beta\mu} \boldsymbol{H}_w$, then there exists $\boldsymbol{H}'_w$ such that $\boldsymbol{H}_w \sim_{w\beta\mu} \boldsymbol{H}'_w$ and $N \rightarrow^*_{w\beta\mu} \boldsymbol{H}'_w$.

Notice that Prop. 1.5 is formulated with respect to '$=_{\beta\mu}$', not '$\sim_{w\beta\mu}$'.

The other two equivalences we consider are generated by *weak head reduction* and *weak explicit head reduction*. We will show in Thm 10.6 that these coincide for pure, substitution-free $\lambda\mu$-terms.

Definition 10.3 (WEAK HEAD EQUIVALENCE) The relation '$\sim_{w\mathrm{H}}$' is defined co-inductively as the largest symmetric binary relation on $\lambda\mu$ such that: $M \sim_{w\mathrm{H}} N$ if and only if either:

- M and N have both no WHNF, or
- both $M \rightarrow^{nf}_{w\mathrm{H}} M'$ and $N \rightarrow^{nf}_{w\mathrm{H}} N'$, and either:
  - if $M' = xM_1 \cdots M_n$ $(n \geq 0)$, then $N' = xN_1 \cdots N_n$ and $M_i \sim_{w\mathrm{H}} N_i$ for all $i \in \underline{n}$; or
  - if $M' = \lambda x.M''$, then $N' = \lambda x.N''$ and $M'' \sim_{w\mathrm{H}} N''$; or
  - if $M' = \mu\alpha.[\beta]M''$, then $N' = \mu\alpha.[\beta]N''$ (so $\alpha \neq \beta$ or $\alpha \in fn(M'')$, $M'' \neq \mu\gamma.[\delta]R$, and similarly for N''), and $M'' \sim_{w\mathrm{H}} N''$.

Notice that $\lambda z.\Delta\Delta \sim_{w\mathrm{H}} \lambda z.\Omega\Omega$ because $\Delta\Delta \sim_{w\mathrm{H}} \Omega\Omega$, since neither has a WHNF.

We perhaps need to clarify the details of this definition. The notion of weak head equivalence captures the fact that, once weak head reduction has finished, there are sub-terms that can be reduced further by themselves. This process can generate infinite terms and the equivalence expresses when it produces equal (infinite) terms. However, it also equates terms that

have no WHNF. As can be seen from Def. 8.4, a context switch $\mu\alpha.[\beta]N$ is in WHNF only if N is; so when we state in the third case that $M \to^{nf}_{w\text{H}} \mu\alpha.[\beta]M''$, by the fact that this reduction has terminated, we know that M'' *is* in WHNF.

We will now define a notion of weak explicit head equivalence, that, in approach, corresponds to weak head equivalence but for the fact that now explicit substitutions are part of terms.

Definition 10.4 (WEAK EXPLICIT HEAD EQUIVALENCE) The relation '$\sim_{w\text{xH}}$' is defined co-inductively as the largest symmetric binary relation on $\lambda\mu\mathbf{x}$ such that: $M \sim_{w\text{xH}} N$ if and only if either:

- M and N have both no WXHNF, or
- both $M \to^{nf}_{w\text{xH}} M'$ and $N \to^{nf}_{w\text{xH}} N'$, and either:
  - if $M' = xM_1\boldsymbol{S}_1 \cdots M_n\boldsymbol{S}_n$ $(n \geq 0)$, then $N' = xN_1\boldsymbol{S}'_1 \cdots N_n\boldsymbol{S}'_n$ (so $x \notin \boldsymbol{S}_i$, $x \notin \boldsymbol{S}'_i$, for $i \in \underline{n}$) and $M_i\boldsymbol{S} \sim_{w\text{xH}} N_i\boldsymbol{S}'$ for all $i \in \underline{n}$ where $\boldsymbol{S} = \boldsymbol{S}_1 \cdots \boldsymbol{S}_n$ and $\boldsymbol{S}' = \boldsymbol{S}'_1 \cdots \boldsymbol{S}'_n$; or
  - if $M' = \lambda x.M''$, then $N' = \lambda x.N''$ and $M'' \sim_{w\text{xH}} N''$; or
  - if $M' = \mu\alpha.[\beta]M''$, then $N' = \mu\alpha.[\beta]N''$ (so $\alpha \neq \beta$ or $\alpha \in fn(M'')$, $M'' \neq \mu\gamma.[\delta]R$, and similarly for N'') and $M'' \sim_{w\text{xH}} N''$.

Notice that $\mu\alpha.[\beta]\Delta\Delta \sim_{w\text{xH}} \Delta\Delta$.

The following results formulate the strong relation between '$\sim_{w\text{H}}$' and '$\sim_{w\text{xH}}$', and therefore between '$\to_{w\text{H}}$' and '$\to_{w\text{xH}}$'. We first show that pure terms that are equivalent under '$\sim_{w\text{xH}}$' are also so under '$\sim_{w\text{H}}$'.

Lemma 10.5 Let $M, N \in \lambda\mu$. $M \sim_{w\text{H}} N$ if and only if there are M', $N' \in \lambda\mu\mathbf{x}$ such that $M' \to^{nf}_{:=} M$ and $N' \to^{nf}_{:=} N$, and $M' \sim_{w\text{xH}} N'$.

Proof *only if*: By coinduction on the definition of '$\sim_{w\text{H}}$'. If $M \sim_{w\text{H}} N$, then either:

- $M \to^{nf}_{w\text{H}} xM_1 \cdots M_n$ and $N \to^{nf}_{w\text{H}} xN_1 \cdots N_n$ and $M_i \sim_{w\text{H}} N_i$, for all $i \in \underline{n}$. Then, by Lemma 8.3, there exist $\overrightarrow{M'_i}$, $\overrightarrow{N'_i}$ such that both
$$\begin{array}{lllll} M & \to^{nf}_{w\text{xH}} & xM'_1\boldsymbol{S}_1 \cdots M'_n\boldsymbol{S}_n & \to^{*}_{:=} & xM_1 \cdots M_n \quad \text{and} \\ N & \to^{nf}_{w\text{xH}} & xN'_1\boldsymbol{S}'_1 \cdots N'_n\boldsymbol{S}'_n & \to^{*}_{:=} & xN_1 \cdots N_n \end{array}$$
Let $\boldsymbol{S} = \boldsymbol{S}_1 \cdots \boldsymbol{S}_n$ and $\boldsymbol{S}' = \boldsymbol{S}'_1 \cdots \boldsymbol{S}'_n$, then in particular $M'_i\boldsymbol{S} \to^{nf}_{:=} M_i$ and $N'_i\boldsymbol{S}' \to^{nf}_{:=} N_i$, for all $i \in \underline{n}$; then by induction, $M'_i\boldsymbol{S} \sim_{w\text{xH}} N'_i\boldsymbol{S}'$ for all $i \in \underline{n}$. But then $M \sim_{w\text{xH}} N$.
- $M \to^{nf}_{w\text{H}} \lambda x.P$, then $N \to^{nf}_{w\text{H}} \lambda x.Q$ and $P \sim_{w\text{H}} Q$. By Lemma 8.3 there exists P' and Q' such that
$$\begin{array}{lllll} M & \to^{nf}_{w\text{xH}} & \lambda x.P' & \to^{nf}_{:=} & \lambda x.P \quad \text{and} \\ N & \to^{nf}_{w\text{xH}} & \lambda x.Q' & \to^{nf}_{:=} & \lambda x.Q \end{array}$$
Then also $P' \to^{nf}_{:=} P$ and $Q' \to^{nf}_{:=} Q$, so by induction $P' \sim_{w\text{xH}} Q'$. But then $M \sim_{w\text{xH}} N$.
- $M \to^{nf}_{w\text{H}} \mu\delta.[\gamma]P$, then $N \to^{nf}_{w\text{H}} \mu\delta.[\gamma]Q$ and $P \sim_{w\text{H}} Q$; similar to the previous part.

The other cases are similar.

if: By coinduction on the definition of '$\sim_{w\text{xH}}$'. If there are M', N' such that $M' \to^{nf}_{:=} M$ and $N' \to^{nf}_{:=} N$, and $M' \sim_{w\text{xH}} N'$, then either:

- $M' \to^{nf}_{w\text{xH}} xM'_1\boldsymbol{S}_1 \cdots M'_n\boldsymbol{S}_n$, $N' \to^{nf}_{w\text{xH}} xN'_1\boldsymbol{S}'_1 \cdots N'_n\boldsymbol{S}'_n$, $\boldsymbol{S} = \boldsymbol{S}_1 \cdots \boldsymbol{S}_n$ and $\boldsymbol{S}' = \boldsymbol{S}'_1 \cdots \boldsymbol{S}'_n$, and $M'_i\boldsymbol{S} \sim_{w\text{xH}} N'_i\boldsymbol{S}'$, for all $i \in \underline{n}$. Let, for all $i \in \underline{n}$, $M'_i\boldsymbol{S} \to^{nf}_{:=} M_i$ and $N'_i\boldsymbol{S}' \to^{nf}_{:=} N_i$ then by induction, $M_i \sim_{w\text{H}} N_i$, for all $i \in \underline{n}$. Let $M' \to^{nf}_{:=} M$; since we have $M' \to^{nf}_{w\text{xH}} xM'_1\boldsymbol{S}_1 \cdots M'_n\boldsymbol{S}_n \to^{nf}_{:=} xM_1 \cdots M_n$, by Lemma 8.3, $M \to^{nf}_{w\text{H}} xM_1 \cdots M_n$. Likewise, we have $N \to^{nf}_{w\text{H}} xN_1 \cdots N_n$. But then $M \sim_{w\text{H}} N$.
- $M' \to^{nf}_{w\text{xH}} \lambda x.P'$, $N' \to^{nf}_{w\text{xH}} \lambda x.Q'$, and $P' \sim_{w\text{xH}} Q'$. Let $P' \to^{nf}_{:=} P$, and $Q' \to^{nf}_{:=} Q$, then by induction, $P \sim_{w\text{H}} Q$. Then we have $M' \to^{nf}_{w\text{xH}} \lambda x.P' \to^{nf}_{:=} \lambda x.P$, and by Lemma 8.3, $M \to^{nf}_{w\text{H}} \lambda x.P$. Similarly, we have $N \to^{nf}_{w\text{H}} \lambda x.Q$; so $M \sim_{w\text{H}} N$.

– $M' \to^{nf}_{w\text{xH}} \mu\delta.[\gamma]P'$, $N' \to^{nf}_{w\text{xH}} \mu\delta.[\gamma]Q'$, and $P' \sim_{w\text{xH}} Q'$; similar to the previous part.

The other cases are similar. □

Notice that this lemma in fact shows:

Corollary 10.6 Let $M, N \in \lambda\mu$, then $M \sim_{w\text{xH}} N \iff M \sim_{w\text{H}} N$.

11 Weak approximation for $\lambda\mu$

In the next section we will show our main result, *i.e.* that the logical encoding is fully abstract with respect to weak equivalence between pure $\lambda\mu$-terms. To achieve this, we show in Thm 12.1 that $\llbracket M \rrbracket^{\mathrm{L}}_{\mathbb{J}} a \approx \llbracket N \rrbracket^{\mathrm{L}}_{\mathbb{J}} a \iff M \sim_{w\text{xH}} N$. To complete the proof towards '$\sim_{w\beta\mu}$', we are thus left with the obligation to show that $M \sim_{w\text{xH}} N \iff M \sim_{w\beta\mu} N$. In Corollary 10.6 we have shown that $M \sim_{w\text{xH}} N \iff M \sim_{w\text{H}} N$, for pure terms; to achieve $M \sim_{w\text{H}} N \iff M \sim_{w\beta\mu} N$, in this section we go through a notion of *weak approximation*. Based on Wadsworth's approach [57], we define '$\sim_{\mathcal{A}_w}$' that expresses that terms have the same weak approximants and show that $M \sim_{w\text{H}} N \iff M \sim_{\mathcal{A}_w} N \iff M \sim_{w\beta\mu} N$.

The notions of *approximation* and *approximant* were first introduced by Wadsworth for the λ-calculus [57], where they are used in order to better express the relation between equivalence of meaning in Scott's models and the usual notions of conversion and reduction. Wadsworth defines approximation of terms through the replacement of any parts of a term remaining to be evaluated (*i.e.* β-redexes) by $\bot$. Repeatedly applying this process over a reduction sequence starting with M gives a set of approximants, each giving some - in general incomplete - information about the result of reducing M. Once this reduction produces a term of the shape $\lambda x_1 \cdots x_m . y N_1 \cdots N_n$ (a head-normal form), all remaining redexes occur in $N_1, \ldots, N_n$, which then in turn will be approximated.

Following this approach, Wadsworth [57] defines $\mathcal{A}(M)$ (similar to Def. 11.1 below) as the set of approximants of the λ-term M, which forms a meet semi-lattice; in [58], the connection is established between approximation and semantics, by showing

$$\llbracket M \rrbracket_{\mathbb{J}\mathrm{D}_\infty} p \;=\; \bigsqcup \{ \llbracket A \rrbracket_{\mathbb{J}\mathrm{D}_\infty} p \mid A \in \mathcal{A}(M) \}.$$

So, essentially, approximants are partially evaluated expressions in which the locations of incomplete evaluation (*i.e.* where reduction *may* still take place) are explicitly marked by the element $\bot$; thus, they *approximate* the result of computations. Intuitively, an approximant can be seen as a 'snapshot' of a reduction, where we focus on that part of the resulting term which will no longer change.

We now define a *weak approximation semantics* for $\lambda\mu$. Approximation for $\lambda\mu$ has been studied by others as well [54, 41, 8]; however, seen that we are mainly interested in *weak* reduction here, we will define *weak* approximants, which are normally not considered.

Definition 11.1 (Weak approximation for $\lambda\mu$) *i)* We define the set of $\lambda\mu\bot$-terms as in Def. 1.1, but add the term constant $\bot$.

$$M, N \;::=\; x \mid \bot \mid \lambda x.M \mid MN \mid \mu\alpha.[\beta]M$$

ii) The set of $\lambda\mu$'s *weak approximants* $\mathcal{A}_w \subseteq \lambda\mu\bot$ with respect to '$\to_{\beta\mu}$' is defined through the grammar:

$$\begin{array}{rcll} \mathbf{A}_w & ::= & \bot & \\ & \mid & x\mathbf{A}^1_w \cdots \mathbf{A}^n_w & (n \geq 0) \\ & \mid & \lambda x.\mathbf{A}_w & \\ & \mid & \mu\alpha.[\beta]\mathbf{A}_w & (\alpha \neq \beta \text{ or } \alpha \in \mathbf{A}_w,\ \mathbf{A}_w \neq \mu\gamma.[\delta]\mathbf{A}'_w,\ \mathbf{A}_w \neq \bot) \end{array}$$

iii) The relation '$\sqsubseteq$' $\subseteq \lambda\mu\bot^2$ is defined as the smallest preorder that is the compatible extension of $\bot \sqsubseteq M$, *i.e.*:

$$\begin{array}{rcl} \bot & \sqsubseteq & M \\ x & \sqsubseteq & x \\ M \sqsubseteq M' & \Rightarrow & \lambda x.M \sqsubseteq \lambda x.M' \;\&\; \mu\gamma.[\delta]M \sqsubseteq \mu\gamma.[\delta]M' \\ M_1 \sqsubseteq M_1' \wedge M_2 \sqsubseteq M_2' & \Rightarrow & M_1M_2 \sqsubseteq M_1'M_2' \end{array}$$

iv) The set of *weak approximants* of $M \in \lambda\mu$, $\mathcal{A}_w(M)$, is defined through:

$$\mathcal{A}_w(M) \;\triangleq\; \{\, \mathbf{A}_w \in \mathcal{A}_w \mid \exists N \in \lambda\mu \,(\; M \to^*_{\beta\mu} N \wedge \mathbf{A}_w \sqsubseteq N) \,\}$$ [19]

v) *Weak approximation equivalence* is defined through: $M \sim_{\mathcal{A}_w} N \;\triangleq\; \mathcal{A}_w(M) = \mathcal{A}_w(N)$.

Notice that if $\mathbf{A}_1 \sqsubseteq M_1$ and $\mathbf{A}_2 \sqsubseteq M_2$, then $\mathbf{A}_1\mathbf{A}_2$ need not be an approximant; it is one if $\mathbf{A}_1 = x\mathbf{A}_1^1 \cdots \mathbf{A}_1^n$, perhaps prefixed with a context switch of the shape $\mu\alpha.[\beta]$. Moreover,

$$\begin{array}{rcccl} \mathcal{A}_w(\lambda z.\Delta\Delta) & = & \{\,\bot, \lambda z.\bot\,\} & = & \mathcal{A}_w(\lambda z.\Omega\Omega) \\ \mathcal{A}_w(\mu\alpha.[\beta]\Delta\Delta) & = & \{\bot\} & = & \mathcal{A}_w(\Delta\Delta) \end{array}$$

Weak approximants are also the normal forms with respect to the notion of reduction on $\lambda\mu\bot$-terms that is the extension of '$\to_{\beta\mu}$' by adding the reduction rules:

$$\begin{array}{rcl} \bot M & \to & \bot \\ \mu\alpha.[\beta]\bot & \to & \bot \end{array}$$

(so not $\lambda x.\bot \to \bot$) but this will play no role in this paper.

The relationship between the approximation relation and reduction is characterised by the following result:

Lemma 11.2 *i) If $\mathbf{A}_w \sqsubseteq M$ and $M \to^*_{\beta\mu} N$, then $\mathbf{A}_w \sqsubseteq N$.*

ii) If $\mathbf{A}_w \in \mathcal{A}_w(N)$ and $M \to^*_{\beta\mu} N$, then also $\mathbf{A}_w \in \mathcal{A}_w(M)$.

iii) If $\mathbf{A}_w \in \mathcal{A}_w(M)$ and $M \to_{\beta\mu} N$, then there exists L such that $N \to^*_{\beta\mu} L$ and $\mathbf{A}_w \sqsubseteq L$.

iv) M is a WHNF *if and only if there exists $\mathbf{A}_w \neq \bot$ such that $\mathbf{A}_w \sqsubseteq M$.*

v) M has no WHNF *if and only if $\mathcal{A}_w(M) = \{\bot\}$.*

Proof: Easy. □

We could have defined the set of approximants of a term coinductively.

Definition 11.3 We define $\mathcal{A}^w(M)$ coinductively by:

- If $\mathbf{A}_w \sqsubseteq M$, then $\mathbf{A}_w \in \mathcal{A}^w(M)$.
- if $M \to^*_{w\mathrm{H}} xM_1 \cdots M_n$ $(n \geq 0)$, then $\mathcal{A}^w(M) = \{\, x\mathbf{A}_w^1 \cdots \mathbf{A}_w^n \mid \forall i \in \underline{n}\; (\mathbf{A}_w^i \in \mathcal{A}^w(M_i)) \,\}$.
- if $M \to^*_{w\mathrm{H}} \lambda x.N$, then $\mathcal{A}^w(M) = \{\, \lambda x.\mathbf{A}_w \mid \mathbf{A}_w \in \mathcal{A}^w(N) \,\}$.
- if $M \to^*_{w\mathrm{H}} \mu\alpha.[\beta]N$, then $\mathcal{A}^w(M) = \{\, \mu\alpha.[\beta]\mathbf{A}_w \mid \mathbf{A}_w \in \mathcal{A}^w(N) \,\}$.

We can show that these definitions coincide:

Lemma 11.4 $\mathcal{A}^w(M) = \mathcal{A}_w(M)$.

Proof:$\subseteq$: If $\mathbf{A}_w \in \mathcal{A}^w(M)$, then by Def. 11.3 either:

[18] For 'normal' approximants, case $\lambda x.\mathbf{A}$ demands that $\mathbf{A} \neq \bot$, as motived by the relation with D_∞. We explicitly drop that restriction here.

[19] Notice that we use '$\to_{\beta\mu}$' here, not '$\to_{w\beta\mu}$'; the approximants are weak, not the reduction.

$\mathbf{A}_w \sqsubseteq M$: Immediate.

$\mathbf{A}_w = x\mathbf{A}_w^1 \cdots \mathbf{A}_w^n$: Then $M \to_{w\mathrm{H}}^* xM_1 \cdots M_n$ for some $M_1, \ldots M_n$, with $\mathbf{A}_w^i \in \mathcal{A}^w(M_i)$, for every $i \in \underline{n}$; by coinduction, also $\mathbf{A}_w^i \in \mathcal{A}_w(M_i)$. Then, by Def. 11.1, for every $i \in \underline{n}$ there exist M_i' such that $M_i \to_{\beta\mu}^* M_i'$ and $\mathbf{A}_w^i \sqsubseteq M_i'$. Since '$\to_{w\mathrm{H}}^*$' $\subseteq$ '$\to_{\beta\mu}^*$', in particular $M \to_{\beta\mu}^* xM_1' \cdots M_n'$; we have $\mathbf{A}_w \sqsubseteq xM_1' \cdots M_n'$, so $\mathbf{A}_w \in \mathcal{A}_w(M)$.

The other cases are similar.

$\supseteq$: If $\mathbf{A}_w \in \mathcal{A}_w(M)$, then by Def. 11.1, there exists N such that $M \to_{\beta\mu}^* N$ and $\mathbf{A}_w^i \sqsubseteq N$. Now either:

$\mathbf{A}_w \sqsubseteq M$: Trivial.

$\mathbf{A}_w = x\mathbf{A}_w^1 \cdots \mathbf{A}_w^n$: Since $x\mathbf{A}_w^1 \cdots \mathbf{A}_w^n \sqsubseteq N$, $N = xN_1 \cdots N_n$ for some $N_1, \ldots, N_n$, and $\mathbf{A}_w^i \sqsubseteq N_i$, for every $i \in \underline{n}$. Then by Def. 11.3, $\mathbf{A}_w^i \in \mathcal{A}^w(N_i)$, for every $i \in \underline{n}$, and by induction, $\mathbf{A}_w^i \in \mathcal{A}_w(N_i)$. By Lemma 8.5, there exist $M_1, \ldots, M_n$ such that $M \to_{w\mathrm{H}}^* xM_1 \cdots M_n \to_{\beta\mu}^* xN_1 \cdots N_n$; so in particular $M_i \to_{\beta\mu}^* N_i$, for every $i \in \underline{n}$. Then by Lemma 11.2, $\mathbf{A}_w^i \in \mathcal{A}_w(M_i)$ and by Def. 11.3, $\mathbf{A}_w \in \mathcal{A}^w(M)$.

The other cases are similar. □

As a consequence, below we will use whichever definition of approximation, $\mathcal{A}^w(M)$ or $\mathcal{A}_w(M)$, is convenient.

As is standard in other settings, interpreting a $M \in \lambda\mu$ through its set of weak approximants $\mathcal{A}_w(M)$ gives a semantics.

Theorem 11.5 (Weak approximation semantics) *If $M =_{\beta\mu} N$, then $M \sim_{\mathcal{A}_w} N$.*

Proof: $M =_{\beta\mu} N \wedge \mathbf{A}_w \in \mathcal{A}_w(M) \Rightarrow M =_{\beta\mu} N \wedge \exists L\ (M \to_{\beta\mu}^* L \wedge \mathbf{A}_w \sqsubseteq L) \Rightarrow$ *(1.5)* $\exists L, K\ (L \to_{\beta\mu}^* K \wedge N \to_{\beta\mu}^* K \wedge \mathbf{A}_w \sqsubseteq L) \Rightarrow$ *(11.2)* $\exists K\ (N \to_{\beta\mu}^* K \wedge \mathbf{A}_w \sqsubseteq K) \Rightarrow \mathbf{A}_w \in \mathcal{A}_w(N)$ □

The reverse implication of this result does not hold, since terms without whnf (which have only $\perp$ as approximant) are not all related by reduction. But we can show the following full abstraction result:

Theorem 11.6 (Full abstraction of '$\sim_{w\beta\mu}$' versus '$\sim_{\mathcal{A}_w}$') *$M \sim_{w\beta\mu} N$ if and only if $M \sim_{\mathcal{A}_w} N$.*

Proof: if: By coinduction on the definition of the set of weak approximants. If $\mathcal{A}_w(M) = \{\perp\} = \mathcal{A}_w(N)$, then by Lemma 11.2 both M and N have no whnf, so $M \sim_{w\beta\mu} N$. Otherwise, either:

$x\mathbf{A}_w^1 \cdots \mathbf{A}_w^n \in \mathcal{A}_w(M) \wedge x\mathbf{A}_w^1 \cdots \mathbf{A}_w^n \in \mathcal{A}_w(N)$: By Def. 11.3 there exists $M_1, \ldots, M_n$ such that $M \to_{w\mathrm{H}}^* xM_1 \cdots M_n$ and $\mathbf{A}_w^i \in \mathcal{A}_w(M_i)$. Likewise, there exist $N_1, \ldots, N_n$ such that $N \to_{w\mathrm{H}}^* xN_1 \cdots N_n$ and $\mathbf{A}_w^i \in \mathcal{A}_w(N_i)$. So, for $i \in \underline{n}$, $\mathcal{A}_w(M_i) = \mathcal{A}_w(N_i)$ and by induction $M_i \sim_{w\beta\mu} N_i$. Since '$\sim_{w\beta\mu}$' is a congruence, also $xM_1 \cdots M_n \sim_{w\beta\mu} xN_1 \cdots N_n$; since '$\sim_{w\beta\mu}$' is closed under reduction '$\to_{w\beta\mu}$', it is also under '$\to_{w\mathrm{H}}$', and we have $M \sim_{w\beta\mu} N$.

The other cases are similar.

only if: As the proof of Thm 11.5, but using Proposition 10.2 rather than 1.5. □

We can also show that weak head equivalence and weak approximation equivalence coincide:

Theorem 11.7 *$M \sim_{w\mathrm{H}} N$ if and only if $M \sim_{\mathcal{A}_w} N$.*

Proof: only if: By coinduction on the definition of '$\sim_{w\mathrm{H}}$'.

M and N have no whnf: Then, by Lemma 11.2, $\mathcal{A}_w(M) = \{\perp\} = \mathcal{A}_w(N)$.

$M \to_{w\mathrm{H}}^* xM_1 \cdots M_n$: Then also $N \to_{w\mathrm{H}}^* xN_1 \cdots N_n$, and $M_i \sim_{w\mathrm{H}} N_i$ for $i \in \underline{n}$, and by coinduction, $M_i \sim_{\mathcal{A}_w} N_i$, so $\mathcal{A}_w(M_i) = \mathcal{A}_w(N_i)$. Then, by Def. 11.3, we have $\mathcal{A}_w(M) = \mathcal{A}_w(N)$.

The other cases are similar.

if: By coinduction on the definition of the set of weak approximants.

$\mathcal{A}_w(M) = \{\bot\} = \mathcal{A}_w(N)$: Then, by Lemma 11.2, both M and N have no WHNF, so $M \sim_{w\text{H}} N$.

$\mathbf{A}_w = x\mathbf{A}_w^1 \cdots \mathbf{A}_w^n$: Then $M \to_{w\text{H}}^* xM_1 \cdots M_n$, and $\mathbf{A}_w^i \in \mathcal{A}_w(M_i)$, for $i \in \underline{n}$. Since $\mathcal{A}_w(M) = \mathcal{A}_w(N)$, also $N \to_{w\text{H}}^* xN_1 \cdots N_n$, with $\mathbf{A}_w^i \in \mathcal{A}_w(N_i)$, so $\mathcal{A}_w(M_i) = \mathcal{A}_w(N_i)$. Then, by coinduction, $M_i \sim_{w\text{H}} N_i$ for every $i \in \underline{n}$, so $M \sim_{w\text{H}} N$.

The other cases are similar. □

Taking '$\sqcup$' as the (partial, compatible) operation of join on terms in $\mathcal{A}_w$ generated by $\bot \sqcup \mathbf{A}_w = \mathbf{A}_w$, we can also define $\llbracket M \rrbracket_{\mathcal{A}_w} = \sqcup\{\, \mathbf{A}_w \mid \mathbf{A}_w \in \mathcal{A}_w(M) \,\}$; then $\llbracket \cdot \rrbracket_{\mathcal{A}_w}$ corresponds to the ($\lambda\mu$-variant of) Lévy-Longo trees, and it becomes easy to show that: $\llbracket M \rrbracket_{\mathcal{A}_w} = \llbracket N \rrbracket_{\mathcal{A}_w} \iff M \sim_{\mathcal{A}_w} N$. We will skip the details here.

Combined with the results shown in the previous section, we can now state that all equivalences coincide:

Corollary 11.8 Let $M, N \in \lambda\mu$, then $M \sim_{w\text{XH}} N \iff M \sim_{w\text{H}} N \iff M \sim_{\mathcal{A}_w} N \iff M \sim_{w\beta\mu} N$.

12 Full abstraction for the logical interpretation

We now come to the main result of this paper, where we show a full abstraction result for our logical interpretation. First we establish the relation between weak explicit head equivalence and weak bisimilarity.

Theorem 12.1 (FULL ABSTRACTION OF '$\approx$' VERSUS '$\sim_{w\text{XH}}$') *For $M, N \in \lambda\mu\mathbf{x}$: $\llbracket M \rrbracket^{\text{L}} a \approx \llbracket N \rrbracket^{\text{L}} a$ if and only if $M \sim_{w\text{XH}} N$.*

Proof. *only if*: By induction on the structure of terms in $\lambda\mu\mathbf{x}$; we distinguish the following cases.

- $\llbracket M \rrbracket^{\text{L}} a$ can never input nor output; then $\llbracket M \rrbracket^{\text{L}} a \approx 0 \approx \llbracket N \rrbracket^{\text{L}} a$. Assume M has a $\to_{w\text{XH}}$-normal form, then by Lemma 8.8, $\llbracket M \rrbracket^{\text{L}} a$ is not weakly bisimilar to 0; therefore, M and N both have no $\to_{w\text{XH}}$-normal form, so $M \sim_{w\text{XH}} N$.
- $\llbracket M \rrbracket^{\text{L}} a \Downarrow \overline{c}$, then by Lemma 8.9,

$$\begin{array}{rcll} \llbracket M \rrbracket^{\text{L}} a \;\approx\; \llbracket \lambda x.M' \rrbracket^{\text{L}} c & \triangleq & (\nu xb)\,(\llbracket M' \rrbracket^{\text{L}} b \mid \overline{c}\langle x,b\rangle) & \\ M & \to_{w\text{XH}}^* & \lambda x.M' & (a = c) \text{ or} \\ M & \to_{w\text{XH}}^{nf} & \mu\alpha.[c]\lambda x.M' & (a \neq c) \end{array}$$

  Since $\llbracket M \rrbracket^{\text{L}} a \approx \llbracket N \rrbracket^{\text{L}} a$, also $\llbracket N \rrbracket^{\text{L}} a \Downarrow \overline{c}$, so

$$\begin{array}{rcll} \llbracket N \rrbracket^{\text{L}} a & \approx & (\nu xb)\,(\llbracket N' \rrbracket^{\text{L}} b \mid \overline{c}\langle x,b\rangle) & \\ N & \to_{w\text{XH}}^* & \lambda x.N' & (a = c) \text{ or} \\ N & \to_{w\text{XH}}^{nf} & \mu\alpha.[c]\lambda x.N' & (a \neq c) \end{array}$$

  Then also $\llbracket M' \rrbracket^{\text{L}} b \approx \llbracket N' \rrbracket^{\text{L}} b$, and by induction, $M' \sim_{w\text{XH}} N'$; so also $M \sim_{w\text{XH}} N$.
- If $\llbracket M \rrbracket^{\text{L}} a \not\Downarrow_o$, but $\llbracket M \rrbracket^{\text{L}} a \Downarrow x$, then by Lemma 8.9,

$$\begin{array}{rcll} \llbracket M \rrbracket^{\text{L}} a & \approx & \llbracket xM_1\boldsymbol{S}_1 \cdots M_n\boldsymbol{S}_n \rrbracket^{\text{L}} a' & \text{as well as} \\ M & \to_{w\text{XH}}^* & xM_1\boldsymbol{S}_1 \cdots M_n\boldsymbol{S}_n & \end{array}$$

  Let $\boldsymbol{S} = \boldsymbol{S}_1 \cdots \boldsymbol{S}_n$, then we have

$$\llbracket xM_1\boldsymbol{S}_1 \cdots M_n\boldsymbol{S}_n \rrbracket^{\text{L}} a' \;\equiv (6.6)\; (\nu\overline{c}\overline{y}\overline{\alpha})\,(x(u).!u{\to}\overline{c_1} \mid \overrightarrow{\llbracket c_i := M_i \cdot c_{i+1} \rrbracket^{\text{L}}} \mid \llbracket \boldsymbol{S} \rrbracket^{\text{L}})$$

  where $c_n = a'$ and

$$\begin{array}{rcl} \llbracket c_i := (M_i)\cdot c_{i+1} \rrbracket^{\text{L}} & \triangleq & !c_i(v,d).(!\overline{v}(w).\llbracket M_i \rrbracket^{\text{L}} w \mid !d{\to}\overline{c_{i+1}}) \\ \llbracket \boldsymbol{S} \rrbracket^{\text{L}} & \equiv & \overrightarrow{\llbracket y := P \rrbracket^{\text{L}}} \mid \overrightarrow{\llbracket \alpha := Q\cdot\beta \rrbracket^{\text{L}}} \\ \llbracket y_j := P_j \rrbracket^{\text{L}} & \triangleq & !\overline{y_j}(w).\llbracket P_j \rrbracket^{\text{L}} w \\ \llbracket \alpha_k := (Q_k)\cdot\beta_k \rrbracket^{\text{L}} & \triangleq & !\alpha_k(v,d).(!\overline{v}(w).\llbracket Q_k \rrbracket^{\text{L}} w \mid !d{\to}\overline{\beta_k}) \end{array}$$

Since $\llbracket M \rrbracket^{\mathrm{L}} a \approx \llbracket N \rrbracket^{\mathrm{L}} a$, again by Lemma 8.9,

$$\begin{array}{rcl} \llbracket N \rrbracket^{\mathrm{L}} a & \approx & \llbracket x N_1 \cdots N_n \boldsymbol{S}' \rrbracket^{\mathrm{L}} a'' \quad \text{and} \\ N & \to^*_{w\mathrm{xH}} & x N_1 \boldsymbol{S}'_1 \cdots N_n \boldsymbol{S}'_n, \end{array}$$

with $\boldsymbol{S}' = \boldsymbol{S}'_1 \cdots \boldsymbol{S}'_n$. Notice that

$$\llbracket x N_1 \boldsymbol{S}'_1 \cdots N_n \boldsymbol{S}'_n \rrbracket^{\mathrm{L}} a'' \equiv (6.6) \ (\nu \overline{y}\vec{\alpha})\,(x(u).!u \to \overline{e_1} \mid \overrightarrow{\llbracket e_i := N_i \cdot e_{i+1} \rrbracket^{\mathrm{L}}} \mid \llbracket \boldsymbol{S}' \rrbracket^{\mathrm{L}})$$

where $e_n = a''$ and

$$\begin{array}{rcl} \llbracket e_i := N_i \cdot e_{i+1} \rrbracket^{\mathrm{L}} & \triangleq & !e_i(v,d).(!\overline{v}(w).\llbracket N_i \rrbracket^{\mathrm{L}} w \mid !d \to \overline{e_{i+1}}) \\ \llbracket \boldsymbol{S}' \rrbracket^{\mathrm{L}} & \equiv & \overrightarrow{\llbracket y := P' \rrbracket^{\mathrm{L}}} \mid \overrightarrow{\llbracket \alpha := Q' \cdot \beta \rrbracket^{\mathrm{L}}} \\ \llbracket y_j := P'_j \rrbracket^{\mathrm{L}} & \triangleq & !\overline{y_j}(w).\llbracket P'_j \rrbracket^{\mathrm{L}} w \\ \llbracket \alpha_k := (Q'_k) \cdot \beta_k \rrbracket^{\mathrm{L}} & \triangleq & !\alpha_k(v,d).(!\overline{v}(w).\llbracket Q'_k \rrbracket^{\mathrm{L}} w \mid !d \to \overline{\beta_k}) \end{array}$$

Since we have

$$\llbracket x M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n \rrbracket^{\mathrm{L}} a' \approx \llbracket x N_1 \boldsymbol{S}'_1 \cdots N_n \boldsymbol{S}'_n \rrbracket^{\mathrm{L}} a'',$$

we infer that $a' = a''$, $c_i = e_i$, and $\llbracket M'_i \boldsymbol{S} \rrbracket^{\mathrm{L}} w \approx \llbracket N'_i \boldsymbol{S}' \rrbracket^{\mathrm{L}} w$ for all $i \in \underline{n}$; then by induction, $M'_i \boldsymbol{S} \sim_{w\mathrm{xH}} N'_i \boldsymbol{S}'$ for all $i \in \underline{n}$, and then also $M \sim_{w\mathrm{xH}} N$.

Notice that the base case for the induction is included in the last part.

if: By coinduction on the definition of '$\sim_{w\mathrm{xH}}$'. Let $M \sim_{w\mathrm{xH}} N$, then either:

- M and N have both no $\to_{w\mathrm{xH}}$-normal form, so, by Lemma 8.7, their interpretations are both weakly bisimilar to the process $\mathit{0}$, so in particular $\llbracket M \rrbracket^{\mathrm{L}} a \approx \llbracket N \rrbracket^{\mathrm{L}} a$; or
- both $M \to^{nf}_{w\mathrm{xH}} M'$ and $N \to^{nf}_{w\mathrm{xH}} N'$, and either:
  * $M' = x M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n$ $(n \geq 0)$, and $N = x N_1 \boldsymbol{S}'_1 \cdots N_n \boldsymbol{S}'_n$, and $M_i \boldsymbol{S} \sim_{w\mathrm{xH}} N_i \boldsymbol{S}'$, for all $i \in \underline{n}$, where $\boldsymbol{S} = \boldsymbol{S}_1 \cdots \boldsymbol{S}_n$, $\boldsymbol{S}' = \boldsymbol{S}'_1 \cdots \boldsymbol{S}'_n$. By Thm 8.6, we know that both $\llbracket M \rrbracket^{\mathrm{L}} a \approx \llbracket x M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n \rrbracket^{\mathrm{L}} a$ and $\llbracket N \rrbracket^{\mathrm{L}} a \approx \llbracket x N_1 \boldsymbol{S}'_1 \cdots N_n \boldsymbol{S}'_n \rrbracket^{\mathrm{L}} a$. Notice that

$$\llbracket x M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n \rrbracket^{\mathrm{L}} a \equiv (6.6) \ (\nu \vec{y}\vec{\alpha}\vec{c})\,(\llbracket x \rrbracket^{\mathrm{L}} c_1 \mid \overrightarrow{\llbracket c_i := M_i \cdot c_{i+1} \rrbracket^{\mathrm{L}}} \mid \llbracket \boldsymbol{S} \rrbracket^{\mathrm{L}})$$

where $c_n = a$ and

$$\begin{array}{rcl} \llbracket x \rrbracket^{\mathrm{L}} c_1 & \triangleq & x(u).!u \to \overline{c_1} \\ \llbracket c_i := (M_i) \cdot c_{i+1} \rrbracket^{\mathrm{L}} & \triangleq & !c_i(v,d).(!\overline{v}(w).\llbracket M_i \rrbracket^{\mathrm{L}} w \mid !d \to \overline{c_{i+1}}) \\ \llbracket \boldsymbol{S} \rrbracket^{\mathrm{L}} & \equiv & \overrightarrow{\llbracket y := P \rrbracket^{\mathrm{L}}} \mid \overrightarrow{\llbracket \alpha := Q \cdot \beta \rrbracket^{\mathrm{L}}} \\ \llbracket y_j := P_j \rrbracket^{\mathrm{L}} & \triangleq & !\overline{y_j}(w).\llbracket P_j \rrbracket^{\mathrm{L}} w \\ \llbracket \alpha_k := (Q_k) \cdot \beta_k \rrbracket^{\mathrm{L}} & \triangleq & !\alpha_k(v,d).(!\overline{v}(w).\llbracket Q_k \rrbracket^{\mathrm{L}} w \mid !d \to \overline{\beta_k}) \end{array}$$

and similarly for $\llbracket x N_1 \boldsymbol{S}'_1 \cdots N_n \boldsymbol{S}'_n \rrbracket^{\mathrm{L}} a$. By induction (see Def. 10.4),

$$(\nu \vec{y}\vec{\alpha})\,(\llbracket M_i \rrbracket^{\mathrm{L}} w \mid \llbracket \boldsymbol{S} \rrbracket^{\mathrm{L}}) \triangleq \llbracket M_i \boldsymbol{S} \rrbracket^{\mathrm{L}} w \approx \llbracket N_i \boldsymbol{S}' \rrbracket^{\mathrm{L}} w \triangleq (\nu \vec{y}\vec{\alpha})\,(\llbracket N_i \rrbracket^{\mathrm{L}} w \mid \llbracket \boldsymbol{S}' \rrbracket^{\mathrm{L}})$$

Then, since '$\approx$' is a congruence, for all $i \in \underline{n}$ also

$$\begin{array}{l} (\nu \vec{y}\vec{\alpha})\,(!c_i(v,d).(!\overline{v}(w).\llbracket M_i \rrbracket^{\mathrm{L}} w \mid !d \to \overline{c_{i+1}}) \mid \llbracket \boldsymbol{S} \rrbracket^{\mathrm{L}}) \approx \\ \qquad (\nu \vec{y}\vec{\alpha})\,(!c_i(v,d).(!\overline{v}(w).\llbracket N_i \rrbracket^{\mathrm{L}} w \mid !d \to \overline{c_{i+1}}) \mid \llbracket \boldsymbol{S}' \rrbracket^{\mathrm{L}}) \end{array}$$

so also $\llbracket x M_1 \boldsymbol{S}_1 \cdots M_n \boldsymbol{S}_n \rrbracket^{\mathrm{L}} a \approx \llbracket x N_1 \boldsymbol{S}'_1 \cdots N_n \boldsymbol{S}'_n \rrbracket^{\mathrm{L}} a$ but then also $\llbracket M \rrbracket^{\mathrm{L}} a \approx \llbracket N \rrbracket^{\mathrm{L}} a$.
  * $M' = \lambda x.M''$, $N' = \lambda x.N''$, and $M'' \sim_{w\mathrm{xH}} N''$. By Thm 8.6, we have $\llbracket M \rrbracket^{\mathrm{L}} a \approx \llbracket \lambda x.M'' \rrbracket^{\mathrm{L}} a$ and $\llbracket N \rrbracket^{\mathrm{L}} a \approx \llbracket \lambda x.N'' \rrbracket^{\mathrm{L}} a$. Notice that

$$\begin{array}{rcl} \llbracket \lambda x.M'' \rrbracket^{\mathrm{L}} a & \triangleq & (\nu x b)\,(\llbracket M'' \rrbracket^{\mathrm{L}} b \mid \overline{a}\langle x,b \rangle) \quad \text{and} \\ \llbracket \lambda x.N'' \rrbracket^{\mathrm{L}} a & \triangleq & (\nu x b)\,(\llbracket N'' \rrbracket^{\mathrm{L}} b \mid \overline{a}\langle x,b \rangle) \end{array}$$

By induction, $\llbracket M'' \rrbracket^{\mathrm{L}} b \approx \llbracket N'' \rrbracket^{\mathrm{L}} b$. As above, since '$\approx$' is a congruence, also $\llbracket M \rrbracket^{\mathrm{L}} a \approx \llbracket N \rrbracket^{\mathrm{L}} a$.

* $M' = \mu\gamma.[\delta]M''$, $N' = \mu\gamma.[\delta]N''$. Then M'' and N'' themselves are in normal form and $M'' \sim_{w\text{XH}} N''$. By Thm 8.6, $\llbracket M \rrbracket^{\text{L}}_{\rrbracket} a \approx \llbracket \mu\gamma.[\delta]M'' \rrbracket^{\text{L}}_{\rrbracket} a$ and $\llbracket N \rrbracket^{\text{L}}_{\rrbracket} a \approx \llbracket \mu\gamma.[\delta]N'' \rrbracket^{\text{L}}_{\rrbracket} a$. Notice that

$$\begin{array}{lllll} \llbracket \mu\gamma.[\delta].M'' \rrbracket^{\text{L}}_{\rrbracket} a & \triangleq & \llbracket M'' \rrbracket^{\text{L}}_{\rrbracket} \delta\{a/\gamma\} & \triangleq & \llbracket M''\{\alpha/\gamma\} \rrbracket^{\text{L}}_{\rrbracket} \delta \quad \text{and} \\ \llbracket \mu\gamma.[\delta].N'' \rrbracket^{\text{L}}_{\rrbracket} a & \triangleq & \llbracket N'' \rrbracket^{\text{L}}_{\rrbracket} \delta\{a/\gamma\} & \triangleq & \llbracket N''\{\alpha/\gamma\} \rrbracket^{\text{L}}_{\rrbracket} \delta \end{array}$$

By induction, $\llbracket M''\{\alpha/\gamma\} \rrbracket^{\text{L}}_{\rrbracket} \delta \approx \llbracket N''\{\alpha/\gamma\} \rrbracket^{\text{L}}_{\rrbracket} \delta$; since '$\approx$' is a congruence, $\llbracket M \rrbracket^{\text{L}}_{\rrbracket} a \approx \llbracket N \rrbracket^{\text{L}}_{\rrbracket} a$. □

We can now prove our main result:

Theorem 12.2 (Full abstraction) *Let $M, N \in \lambda\mu$, then $\llbracket M \rrbracket^{\text{L}}_{\rrbracket} a \approx \llbracket N \rrbracket^{\text{L}}_{\rrbracket} a$ if and only if $M \sim_{w\beta\mu} N$.*

Proof: By Corollary 11.8 and Thm 12.1. □

Conclusions

We have defined $\lambda\mu\mathbf{x}$, a variant of $\lambda\mu$ that uses explicit substitution, and defined a notion of explicit head reduction '$\rightarrow_{\text{XH}}$' that only works on the head of a term, so only ever replaces the head variable of a term. We have found a new, simple and intuitive interpretation of $\lambda\mu\mathbf{x}$-terms in π that uses the naming mechanism of $\lambda\mu$ and gives a name to the anonymous output of terms and respects '$\rightarrow_{\text{XH}}$'. For this interpretation, we have shown that termination is preserved, and that it is sound and complete, as well as that it gives a semantics for $\lambda\mu\mathbf{x}$ and for $\lambda\mu$.

We also defined a weak variant of explicit head reduction, '$\rightarrow_{w\text{XH}}$'. This naturally leads to a notion of weak head normal form and weak approximation and we have shown that interpreting a term by the set of its weak approximants gives a semantics for $\lambda\mu$ as well. We have defined the weak equivalences '$\sim_{w\beta\mu}$', '$\sim_{w\text{H}}$', '$\sim_{w\text{XH}}$', and '$\sim_{\mathcal{A}_w}$' on $\lambda\mu$ terms, and have shown that these all coincide on *pure* terms (without explicit substitution). We have proven that $M \sim_{w\text{XH}} N \iff \llbracket M \rrbracket^{\text{L}}_{\rrbracket} a \approx \llbracket N \rrbracket^{\text{L}}_{\rrbracket} a$, which, combined with our other results, shows that our interpretation is fully abstract with respect to weak equivalences on terms.

Acknowledgements

We are greatly indebted to the anonymous referees whose comments and corrections have greatly improved our paper. We like to thank Nobuko Yoshida for useful discussions.

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit Substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[2] M. Abadi and A. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. In *Proceedings of the Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.

[3] S. Abramsky. The lazy lambda calculus. In *Research topics in functional programming*, pages 65–116. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA, 1990.

[4] S. Abramsky and C.-H.L. Ong. Full Abstraction in the Lazy Lambda Calculus. *Information and Computation*, 105(2):159–267, 1993.

[5] Z.M. Ariola and H. Herbelin. Minimal Classical Logic and Control Operators. In J.C.M. Baeten, J.K. Lenstra, J. Parrow, and G.J. Woeginger, editors, *Proceedings of Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003*, volume 2719 of *Lecture Notes in Computer Science*, pages 871–885. Springer Verlag, 2003.

[6] P. Audebaud. Explicit Substitutions for the Lambda-Mu Calculus. Research Report 94-26, École Normale Supérieure de Lyon, 1994.

[7] S. van Bakel. Strict intersection types for the Lambda Calculus. *ACM Computing Surveys*, 43:20:1–20:49, April 2011.

[8] S. van Bakel. Characterisation of Normalisation Properties for $\lambda\mu$ using Strict Negated Intersection Types. *ACM Transactions on Computational Logic*, 19, 2018.

[9] S. van Bakel, F. Barbanera, and U. de'Liguoro. Intersection Types for the $\lambda\mu$-calculus. *Logical Methods in Computer Science*, 141(1), 2018.

[10] S. van Bakel, L. Cardelli, and M.G. Vigliotti. From $\mathcal{X}$ to π; Representing the Classical Sequent Calculus in the π-calculus. In *Electronic Proceedings of International Workshop on Classical Logic and Computation 2008* (CL&C'08), *Reykjavik, Iceland*, 2008.

[11] S. van Bakel and P. Lescanne. Computation with Classical Sequents. *Mathematical Structures in Computer Science*, 18:555–609, 2008.

[12] S. van Bakel and M.G. Vigliotti. A logical interpretation of the λ-calculus into the π-calculus, preserving spine reduction and types. In M. Bravetti and G. Zavattaro, editors, *Proceedings of 20th International Conference on Concurrency Theory* (CONCUR'09), Bologna, Italy, volume 5710 of *Lecture Notes in Computer Science*, pages 84 – 98. Springer Verlag, 2009.

[13] S. van Bakel and M.G. Vigliotti. An Output-Based Semantics of $\lambda\mu$ with Explicit Substitution in the π-calculus - Extended Abstract. In J.C. M. Baeten, T. Ball, and F.S. de Boer, editors, *Theoretical Computer Science - 7th IFIP TC 1/WG 2.2 International Conference (TCS 2012)*, volume 7604 of *Lecture Notes in Computer Science*, pages 372–387. Springer Verlag, 2012.

[14] S. van Bakel and M.G. Vigliotti. A fully abstract semantics of $\lambda\mu$ in the π-calculus. In *Proceedings of Sixth International Workshop on Classical Logic and Computation 2014* (CL&C'14), *Vienna, Austria*, volume 164 of *Electronic Proceedings in Theoretical Computer Science*, pages 33–47, 2014.

[15] H. Barendregt. *The Lambda Calculus: its Syntax and Semantics*. North-Holland, Amsterdam, revised edition, 1984.

[16] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *Journal of Symbolic Logic*, 48(4):931–940, 1983.

[17] E. Beffara and V. Mogbil. Proofs as Executions. In J.C. M. Baeten, T. Ball, and F.S. de Boer, editors, *Theoretical Computer Science - 7th IFIP TC 1/WG 2.2 International Conference (TCS 2012)*, volume 7604 of *Lecture Notes in Computer Science*, pages 280–294. Springer Verlag, 2012.

[18] Emmanuel Beffara. A Concurrent Model for Linear Logic. *Electronic Notes in Theoretical Computer Science*, 155:147–168, 2006.

[19] G. Bellin and P.J. Scott. On the pi-Calculus and Linear Logic. *Theoretical Computer Science*, 135(1):11–65, 1994.

[20] R. Bloo and K.H. Rose. Preservation of Strong Normalisation in Named Lambda Calculi with Explicit Substitution and Garbage Collection. In *CSN'95 – Computer Science in the Netherlands*, pages 62–72, 1995.

[21] G. Boudol and C. Laneve. λ-Calculus, Multiplicities, and the π-Calculus. In G.D. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, pages 659–690. The MIT Press, 2000.

[22] N.G. de Bruijn. Lambda Calculus Notation with Nameless Dummies: A Tool for Automatic Formula Manipulation, with Application to the Church-Rosser Theorem. *Indagationes Mathematicae*, 34:381–392, 1972.

[23] L. Caires and F. Pfenning. Session Types as Intuitionistic Linear Propositions. In P. Gastin and F. Laroussinie, editors, *Concurrency Theory, 21th International Conference, (CONCUR'10), Paris, France, 2010*, volume 6269 of *Lecture Notes in Computer Science*, pages 222–236. Springer Verlag, 2010.

[24] A. Church. A Note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1(1):40–41, 1936.

[25] M. Cimini, C. Sacerdoti Coen, and D. Sangiorgi. Functions as Processes: Termination and the $\lambda\mu\tilde{\mu}$-Calculus. In M. Wirsing, M. Hofmann, and A. Rauschmayer, editors, *Trustworthly Global Computing - 5th International Symposium, TGC 2010, Munich, Germany, February 24-26, 2010, Revised Selected Papers*, volume 6084 of *Lecture Notes in Computer Science*, pages 73–86. Springer Verlag, 2010.

[26] P.-L. Curien and H. Herbelin. The Duality of Computation. In *Proceedings of the 5th ACM SIGPLAN International Conference on Functional Programming (ICFP'00)*, volume 35.9 of *ACM Sigplan Notices*, pages 233–243. ACM, 2000.

[27] J. Engelfriet. A Multiset Semantics for the pi-Calculus with Replication. *Theoretical Computer Science*, 153(1&2):65–94, 1996.

[28] M. Felleisen. *The Calculi of λ-v-CS Conversion: A Syntactic Theory of Control and State in Imperative Higher-Order Programming Languages*. PhD thesis, Department of Computer Science, Indiana University, Bloomington, Indiana, August 1987.

[29] M. Felleisen, D.P. Friedman, E. Kohlbecker, and B. Duba. Reasoning with Continuations. In *Proceedings of the First Symposium on Logic in Computer Science*, pages 131–141, Cambridge, Massachusetts, June 1986. IEEE.

[30] G. Gentzen. Investigations into logical deduction. In M.E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*. North Holland, 68ff (1969), 1935.
[31] J.-Y Girard. Linear Logic. *Theoretical Computer Science*, 50:1–102, 1987.
[32] Ph. de Groote. On the Relation between the $\lambda\mu$-Calculus and the Syntactic Theory of Sequential Control. In *Proceedings of 5th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'94)*, volume 822 of *Lecture Notes in Computer Science*, pages 31–43. Springer Verlag, 1994.
[33] H. Herbelin. On the Degeneracy of Sigma-Types in Presence of Computational Classical Logic. In P. Urzyczyn, editor, *Typed Lambda Calculi and Applications, 7th International Conference, TLCA 2005, Nara, Japan, April 21-23, 2005, Proceedings*, volume 3461 of *Lecture Notes in Computer Science*, pages 209–220. Springer, 2005.
[34] D. Hirschkoff, J.-M. Madiot, and D. Sangiorgi. Duality and i/o-Types in the -Calculus. In M. Koutny and I. Ulidowski, editors, *Concurrency Theory - 23rd International Conference, (CONCUR 2012)*, volume 7454 of *Lecture Notes in Computer Science*, pages 302–316. Springer, 2012.
[35] K. Honda and M. Tokoro. An Object Calculus for Asynchronous Communication. In Pierre America, editor, *ECOOP'91 European Conference on Object-Oriented Programming, Geneva, Switzerland, Proceedings*, volume 512 of *Lecture Notes in Computer Science*, pages 133–147. Springer Verlag, 1991.
[36] K. Honda, N. Yoshida, and M. Berger. Control in the π-Calculus. In *Proceedings of Fourth ACM-SIGPLAN Continuation Workshop* (CW'04), 2004.
[37] J.W. Klop. Term Rewriting Systems. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 1, pages 1–116. Clarendon Press, 1992.
[38] J-L. Krivine. A call-by-name lambda-calculus machine. *Higher Order and Symbolic Computation*, 20(3):199–207, 2007.
[39] S.B. Lassen. Head Normal Form Bisimulation for Pairs and the $\lambda\mu$-Calculus. In *Proceedings of 21th IEEE Symposium on Logic in Computer Science (LICS'06),Seattle, WA, USA*, pages 297–306, 2006.
[40] J.-J Lévy. An Algebraic Interpretation of the $\lambda\beta\mathbf{K}$-calculus and an Application of a Labelled λ-calculus. *Theoretical Computer Science*, 2(1):97–114, 1976.
[41] U. de'Liguoro. The Approximation Theorem for the $\Lambda\mu$-Calculus. *Mathematical Structures in Computer Science*, FirstView:1–21, 2016.
[42] G. Longo. Set-theoretical models of λ-calculus: theories, expansions and isomorphisms. *Annals of Pure and Applied Logic*, 24(2):153–188, 1983.
[43] R. Milner. Functions as Processes. *Mathematical Structures in Computer Science*, 2(2):269–310, 1992.
[44] R. Milner. The Polyadic π-Calculus: A Tutorial. In F.L Bauer, W. Brauer, and H Schwichtenberg, editors, *Logic and Algebra of Specification*. Springer Verlag, Secaucus, NJ, USA, 1993.
[45] R. Milner. *Communicating and Mobile Systems: the π-calculus*. Cambridge University Press, 1999.
[46] C.-H.L. Ong. Fully Abstract Models of the Lazy Lambda Calculus. In *29th Annual Symposium on Foundations of Computer Science*, pages 368–376. IEEE Computer Society, 1988.
[47] C.-H.L. Ong and C.A. Stewart. A Curry-Howard foundation for functional computation with control. In *Proceedings of the 24th Annual ACM Symposium on Principles Of Programming Languages*, pages 215–227, 1997.
[48] M. Parigot. An algorithmic interpretation of classical natural deduction. In *Proceedings of 3rd International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'92)*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer Verlag, 1992.
[49] M. Parigot. Strong Normalization for Second Order Classical Natural Deduction. In *Proceedings of Eighth Annual IEEE Symposium on Logic in Computer Science, Montreal, Canada*, pages 39–46, 1993.
[50] W. Py. *Confluence en $\lambda\mu$-calcul*. Thèse de doctorat, Université de Savoie, 1998.
[51] D. Sangiorgi. The Lazy Lambda Calculus in a Concurrency Scenario. *Information and Computation*, 111(1):120–153, 1994.
[52] D. Sangiorgi and D. Walker. *The Pi-Calculus*. Cambridge University Press, 2001.
[53] A. Saurin. Separation with streams in the $\Lambda\mu$-calculus. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*, pages 356–365, 2005.
[54] A. Saurin. Standardization and Böhm Trees for $\lambda\mu$-calculus. In M. Blume, N. Kobayashi, and G. Vidal, editors, *Functional and Logic Programming, 10th International Symposium, (FLOPS'10), Sendai, Japan*, volume 6009 of *Lecture Notes in Computer Science*, pages 134–149. Springer Verlag, 2010.
[55] Th. Streicher and B. Reus. Classical logic: Continuation Semantics and Abstract Machines. *Journal of Functional Programming*, 11(6):543–572, 1998.
[56] H. Thielecke. *Categorical Structure of Continuation Passing Style*. PhD thesis, University of Edinburgh, 1997. LFCS technical report ECS-LFCS-97-376.
[57] C.P. Wadsworth. The Relation Between Computational and Denotational Properties for Scott's D_∞-Models of the Lambda-Calculus. *SIAM Journal on Computing*, 5(3):488–521, 1976.

[58] C.P. Wadsworth. Approximate Reduction and Lambda Calculus Models. *SIAM Journal on Computing*, 7(3):337–356, 1978.

Appendix A Restricting the interpretation to the λ-calculus

Most of the results shown in this paper hold for the λ-calculus as well, even if they would not follow from the results shown here, but would need to be shown independently (with almost identical proofs). However, some results are formulated here using '$\approx_{\mathrm{D}}$', *i.e.* Lemma 2.5:2, which explicitly deals with the modelling of the distribution of the encoding of the explicit context substitution. In particular, we need '$\approx_{\mathrm{D}}$' in the proof of Thm 7.1 (only) to show that the encoding models the reduction steps (where N gets distributed):

$$\begin{array}{lrcl} (\mu_p): & (\mu\alpha.[\alpha]M)N & \to & \mu\gamma.[\gamma]M\langle\alpha{:=}N{\cdot}\gamma\rangle N \\ (hn): & (\mu\alpha.[\beta]M)\boldsymbol{S} & \to & (\mu\alpha.[\gamma]M\langle\beta{:=}N{\cdot}\gamma\rangle N)\boldsymbol{S}\backslash\beta \quad (\langle\beta{:=}N{\cdot}\gamma\rangle \in \boldsymbol{S}). \end{array}$$

Since these steps are necessary for $\lambda\mu$ only, it is fair to ask if, when restricting to the λ-calculus and explicit head-reduction for $\lambda\mathbf{x}$, the λ-calculus with explicit substitution [20], the formulation of the results can be strengthened. We will briefly discuss that in this section.

First we present $\lambda\mathbf{x}$, Bloo and Rose's [20] λ-calculus with explicit substitution, defined by:

Definition A.1 (Explicit λ-calculus $\lambda\mathbf{x}$ cf. [20]) *i*) The syntax of $\lambda\mathbf{x}$ is defined by:

$$M,N \ ::= \ x \mid \lambda x.M \mid MN \mid M\langle x{:=}N\rangle$$

ii) The reduction relation '$\to_{\mathrm{x}}$' on terms in $\lambda\mathbf{x}$ is defined by the rules:

$$\begin{array}{rcl} (\lambda x.M)N & \to & M\langle x{:=}N\rangle \\ (\lambda y.M)\langle x{:=}L\rangle & \to & \lambda y.(M\langle x{:=}L\rangle) \\ (MN)\langle x{:=}L\rangle & \to & (M\langle x{:=}L\rangle)(N\langle x{:=}L\rangle) \\ x\langle x{:=}L\rangle & \to & L \\ M\langle x{:=}L\rangle & \to & M \quad (x \notin fv(M)) \end{array} \qquad M \to N \ \Rightarrow \ \left\{ \begin{array}{rcl} ML & \to & NL \\ LM & \to & LN \\ \lambda x.M & \to & \lambda x.N \\ M\langle x{:=}L\rangle & \to & N\langle x{:=}L\rangle \\ L\langle x{:=}M\rangle & \to & L\langle x{:=}N\rangle \end{array} \right.$$

Definition A.2 (Explicit head and lazy reduction) *i*) *Explicit head-reduction* '$\to_{\mathrm{xH}}$' on $\lambda\mathbf{x}$ is defined by:

$$\begin{array}{lrcll} (\beta): & (\lambda x.M)N & \to & M\langle x{:=}N\rangle & \\ (hv): & x\boldsymbol{S}_0M_1\boldsymbol{S}_1\cdots M_n\boldsymbol{S}_n\langle x{:=}N\rangle & \to & N\boldsymbol{S}_0M_1\boldsymbol{S}_1\cdots M_n\boldsymbol{S}_n\langle x{:=}N\rangle & (n \geq 0) \\ (\lambda\boldsymbol{S}): & (\lambda y.M)\boldsymbol{S} & \to & \lambda y.(M\boldsymbol{S}) & \\ (gc): & M\boldsymbol{S} & \to & M\boldsymbol{S}\backslash x & (x \in \boldsymbol{S}, x \notin M) \end{array}$$

$$M \to N \ \Rightarrow \ \left\{ \begin{array}{lcl} \lambda x.M & \to & \lambda x.N \\ ML & \to & NL \\ M\boldsymbol{S} & \to & N\boldsymbol{S} \end{array} \right.$$

ii) We define *explicit lazy reduction*[20] '$\to_{\mathrm{xL}}$' by eliminating, from '$\to_{\mathrm{xH}}$', the rules

$$\begin{array}{lrcl} (\lambda\boldsymbol{S}): & (\lambda y.M)\boldsymbol{S} & \to & \lambda y.(M\boldsymbol{S}) \\ & M \to N & \Rightarrow & \lambda x.M \to \lambda x.N \end{array}$$

Notice that we do not allow substitutions to be propagated under abstractions.

As suggested in Sect. 3, we can reformulate Milner's first result (Thm 3.3), in the form that Milner perhaps intended, by showing that his encoding respects explicit lazy reduction, modulo garbage collection.

[20] We could have used the terminology 'call by name' for this notion, but prefer to stick with Milner's choice.

Definition A.3 We extend Milner's interpretation (see Def. 3.1) to $\lambda\mathbf{x}$ by adding the case:

$$\llbracket M\langle x{:=}N\rangle \rrbracket^{\mathrm{M}}_{\rrbracket} a \;\triangleq\; (\nu x)\,(\llbracket M \rrbracket^{\mathrm{M}}_{\rrbracket} a \mid \llbracket x{:=}N \rrbracket^{\mathrm{M}}_{\rrbracket})$$

We can show that Milner's encoding respects single step $\to_{\mathrm{XL}}$-reduction.

Theorem A.4 ($\llbracket\cdot\rrbracket^{\mathrm{M}}_{\rrbracket}\cdot$ PRESERVES $\to_{\mathrm{XL}}$) *If $M \to^*_{\mathrm{XL}} N$, then $\llbracket M \rrbracket^{\mathrm{M}}_{\rrbracket} a \to^*_{\pi},\approx_{\mathrm{G}} \llbracket N \rrbracket^{\mathrm{M}}_{\rrbracket} a$.*

Proof: By induction on the definition of single step explicit lazy reduction; we only show the base cases.

$$\begin{array}{lll}
(\beta)\colon \llbracket(\lambda x.M)N\rrbracket^{\mathrm{M}}_{\rrbracket} a \;\to^+_{\pi}\; (3.2) & (\nu x)\,(\llbracket M \rrbracket^{\mathrm{M}}_{\rrbracket} a \mid \llbracket x{:=}N \rrbracket^{\mathrm{M}}_{\rrbracket}) \;\triangleq\; \llbracket M\langle x{:=}N\rangle \rrbracket^{\mathrm{M}}_{\rrbracket} a & \\
(hv)\colon \llbracket x\boldsymbol{S}_0 M_1\boldsymbol{S}_1\cdots M_n\boldsymbol{S}_n\langle x{:=}N\rangle \rrbracket^{\mathrm{M}}_{\rrbracket} a & & \triangleq \\
\quad (\nu x)\,(\llbracket N\boldsymbol{S}_0 M_1\boldsymbol{S}_1\cdots M_{n-1}\boldsymbol{S}_{n-1}\rrbracket^{\mathrm{M}}_{\rrbracket} a \mid !\,x(w).\llbracket N \rrbracket^{\mathrm{M}}_{\rrbracket} w) & & \triangleq,\equiv \\
\quad (\nu x)\,((\nu c_n\vec{y}_n)\,(\llbracket N\boldsymbol{S}_0 M_1\boldsymbol{S}_1\cdots M_{n-1}\boldsymbol{S}_{n-1}\rrbracket^{\mathrm{M}}_{\rrbracket} c_n & & \\
\qquad\qquad \mid (\nu z)\,(\overline{c_n}\,z.\overline{c_n}\,a.\llbracket z{:=}M_n \rrbracket^{\mathrm{M}}_{\rrbracket}) \mid \llbracket \boldsymbol{S}_n \rrbracket^{\mathrm{M}}_{\rrbracket}) \mid !\,x(w).\llbracket N \rrbracket^{\mathrm{M}}_{\rrbracket} w) & & \triangleq,\equiv \\
\quad (\nu x)\,((\nu\vec{c_i}\vec{\vec{y}_i})\,(\llbracket x \rrbracket^{\mathrm{M}}_{\rrbracket} c_1 \mid \overrightarrow{(\nu z)\,(\overline{c_i}\,z.\overline{c_i}\,c_{i+1}.\llbracket z{:=}M_i \rrbracket^{\mathrm{M}}_{\rrbracket})} \mid \overrightarrow{\llbracket \boldsymbol{S}_i \rrbracket^{\mathrm{M}}_{\rrbracket}}) \mid !\,x(w).\llbracket N \rrbracket^{\mathrm{M}}_{\rrbracket} w) & & \equiv,\triangleq\; (c_{n+1}=a) \\
\quad (\nu x)\,((\nu\vec{c_i}\vec{\vec{y}_i})\,(\overline{x}\,c_1 \mid x(w).\llbracket N \rrbracket^{\mathrm{M}}_{\rrbracket} w \mid \overrightarrow{(\nu z)\,(\overline{c_i}\,z.\overline{c_i}\,c_{i+1}.\llbracket z{:=}M_i \rrbracket^{\mathrm{M}}_{\rrbracket})} & & \\
\qquad\qquad \mid \overrightarrow{\llbracket \boldsymbol{S}_i \rrbracket^{\mathrm{M}}_{\rrbracket}} \mid !\,x(w).\llbracket N \rrbracket^{\mathrm{M}}_{\rrbracket} w)) & & \to_{\pi}\; (x) \\
\quad (\nu x)\,((\nu\vec{c_i}\vec{\vec{y}_i})\,(\llbracket N \rrbracket^{\mathrm{M}}_{\rrbracket} c_1 \mid \overrightarrow{(\nu z)\,(\overline{c_i}\,z.\overline{c_i}\,c_{i+1}.\llbracket z{:=}M_i \rrbracket^{\mathrm{M}}_{\rrbracket})} \mid \overrightarrow{\llbracket \boldsymbol{S}_i \rrbracket^{\mathrm{M}}_{\rrbracket}} \mid !\,x(w).\llbracket N \rrbracket^{\mathrm{M}}_{\rrbracket} w)) & & \triangleq \\
\quad \llbracket N\boldsymbol{S}_0 M_1\boldsymbol{S}_1\cdots M_n\boldsymbol{S}_n\langle x{:=}N\rangle \rrbracket^{\mathrm{M}}_{\rrbracket} a & & \\
(gc)\colon \llbracket M\boldsymbol{S} \rrbracket^{\mathrm{M}}_{\rrbracket} a \;\triangleq\; (\nu\vec{y})\,(\llbracket M \rrbracket^{\mathrm{M}}_{\rrbracket} a \mid \llbracket \boldsymbol{S} \rrbracket^{\mathrm{M}}_{\rrbracket}) & = (\nu\vec{y})\,(\llbracket M \rrbracket^{\mathrm{M}}_{\rrbracket} a \mid \llbracket \boldsymbol{S}\backslash x \rrbracket^{\mathrm{M}}_{\rrbracket} \mid \llbracket \boldsymbol{S}_x \rrbracket^{\mathrm{M}}_{\rrbracket}) & \equiv\; (x \notin fv(M)) \\
\quad (\nu\vec{y}\backslash x)\,(\llbracket M \rrbracket^{\mathrm{M}}_{\rrbracket} a \mid \llbracket \boldsymbol{S}\backslash x \rrbracket^{\mathrm{M}}_{\rrbracket}) \mid (\nu x)\,(\llbracket \boldsymbol{S}_x \rrbracket^{\mathrm{M}}_{\rrbracket}) & \approx_{\mathrm{G}} (\nu\vec{y}\backslash x)\,(\llbracket M \rrbracket^{\mathrm{M}}_{\rrbracket} a \mid \llbracket \boldsymbol{S}\backslash x \rrbracket^{\mathrm{M}}_{\rrbracket}) & \triangleq\; \llbracket M\boldsymbol{S}\backslash x \rrbracket^{\mathrm{M}}_{\rrbracket} a
\end{array}$$

□

Notice that we have shown this for single-step reduction, not just reduction to normal form, and for all terms, not just closed terms.

With this, we can now restate Milner's result:

Corollary A.5 If M is closed, and $M \to^{nf}_{\mathrm{XL}} (\lambda y.R)\,\langle\overrightarrow{x{:=}N}\rangle$, then

$$\llbracket M \rrbracket^{\mathrm{M}}_{\rrbracket} a \;\to^{nf}_{\pi},\approx_{\mathrm{G}}\; (\nu\vec{x})\,(\llbracket \lambda x.R \rrbracket^{\mathrm{M}}_{\rrbracket} a \mid \overrightarrow{\llbracket x := N \rrbracket^{\mathrm{M}}_{\rrbracket}})$$

The restriction of our encoding $\llbracket\cdot\rrbracket^{\mathrm{L}}_{\rrbracket}\cdot$ from Def. 6.1 to $\lambda\mathbf{x}$ is defined by:

Definition A.6 (OUTPUT-BASED ENCODING OF $\lambda\mathbf{x}$-TERMS IN π) The mapping $\llbracket\cdot\rrbracket^{\lambda}_{\rrbracket}\cdot$ is defined by:

$$\begin{array}{rcll}
\llbracket x \rrbracket^{\lambda}_{\rrbracket} a & \triangleq & x(u).u{\to}\overline{a} & (x \neq a) \\
\llbracket \lambda x.M \rrbracket^{\lambda}_{\rrbracket} a & \triangleq & (\nu xb)\,(\llbracket M \rrbracket^{\lambda}_{\rrbracket} b \mid \overline{a}\langle x,b\rangle) & (a,b\ \mathit{fresh}) \\
\llbracket MN \rrbracket^{\lambda}_{\rrbracket} a & \triangleq & (\nu c)\,(\llbracket M \rrbracket^{\lambda}_{\rrbracket} c \mid c(v,d).(\llbracket v := N \rrbracket^{\lambda}_{\rrbracket} \mid d{\to}\overline{a})) & (a,b,c,v,d\ \mathit{fresh}) \\
\llbracket M\langle x{:=}N\rangle \rrbracket^{\lambda}_{\rrbracket} a & \triangleq & (\nu x)\,(\llbracket M \rrbracket^{\lambda}_{\rrbracket} a \mid \llbracket x := N \rrbracket^{\lambda}_{\rrbracket}) & (a\ \mathit{fresh}) \\
\llbracket x := N \rrbracket^{\lambda}_{\rrbracket} & \triangleq & !\,\overline{x}(w).\llbracket N \rrbracket^{\lambda}_{\rrbracket} w & (w\ \mathit{fresh})
\end{array}$$

Notice the absence of replication, compared to the definition of $\llbracket\cdot\rrbracket^{\mathrm{L}}_{\rrbracket}\cdot$, in the cases for variables and application; the main reason for this is that, unlike terms of $\lambda\mu\mathbf{x}$, interpreted $\lambda\mathbf{x}$-terms can only output on the name under which they are interpreted; for example, in the proof for Thm 12.1, when considering $\llbracket M \rrbracket^{\lambda}_{\rrbracket} a \Downarrow \overline{c}$, then $a = c$. Those replications dealt, in particular, with the multi-output character of $\lambda\mu\mathbf{x}$-terms, and are no longer needed; they could be reintroduced, however, without any negative effect. The only (crucial) use of replication remains in the interpretation of explicit substitution, modelling its distributive character.

Notice also the difference in the interpretation of explicit substitution with the one defined in [12], which uses $\llbracket x := N \rrbracket^{\mathrm{s}}_{\rrbracket} a \;\triangleq\; !\llbracket N \rrbracket^{\mathrm{s}}_{\rrbracket} x$.

We can show the same results for $\lambda\mathbf{x}$ and the λ-calculus as those we have shown above for $\lambda\mu\mathbf{x}$ and $\lambda\mu$ in much the same manner, but can remove the use of '$\approx_{\mathrm{D}}$', so Lemma 2.5:2 is not needed. Of course, these new results cannot be direct consequences of the results we have shown for the latter two, since it could be that the presence of μ plays an important role when dealing with the interpretation of $\lambda\mathbf{x}$-terms. However, it is straightforward to verify that

this is not the case; we can copy over all the proofs given above, remove the cases dealing with context switches $\mu\alpha.[\beta]$ and find ourselves with proofs directly for $\lambda\mathbf{x}$. Sometimes the proof gets even more simple; for example, since the interpretation of an application is defined without replication for the context substitution, less garbage needs to be collected during reduction inside interpreted terms.

Similarly, as in Thm 9.2, also for our encoding we can show:

Theorem A.7 *If M is a closed λ-term, and $M \to^{nf}_{\mathrm{xH}} N$ then $\llbracket M \rrbracket^{\lambda} a \to^{nf}_{\pi}, \approx_{\mathrm{G}} \llbracket N \rrbracket^{\lambda} a$.*

so our interpretation follows weak explicit head reduction on closed λ-terms to normal form step by step. We hereby emulate Milner's original result, Thm 3.3, but for the fact that our result is stated with explicit head reduction.

As in Thm 7.1, 7.5, 7.6, 7.7, 7.9, and 7.10 we can show that:

Theorem A.8 *i) If $M \in \lambda\mathbf{x}$, and $M \to_{\mathrm{xH}} N$, then $\llbracket M \rrbracket^{\lambda} a \to^{*}_{\pi}, \approx_{\mathrm{R}}, \approx_{\mathrm{G}} \llbracket N \rrbracket^{\lambda} a$.*

ii) If $M \in \lambda\mathbf{x}$, and $\llbracket M \rrbracket^{\lambda} a \to_{\pi} P$, then there exists N such that $P \approx_{\mathrm{RG}} \llbracket N \rrbracket^{\lambda} a$, and $M \to_{\mathrm{xH}} N$.

iii) If $M, N \in \lambda\mathbf{x}$, and $M =_{\mathrm{x}} N$, then $\llbracket M \rrbracket^{\lambda} a \approx \llbracket N \rrbracket^{\lambda} a$.

iv) If $M, N \in \lambda$, and $M =_{\beta} N$, then $\llbracket M \rrbracket^{\lambda} a \approx \llbracket N \rrbracket^{\lambda} a$.

v) If $M \in \lambda\mathbf{x}$, and $M \to^{nf}_{\mathrm{xH}} N$, then $\llbracket M \rrbracket^{\lambda} a \Downarrow_{\pi}$.

vi) If $M \in \lambda$, and $M \to^{hnf}_{\beta} N$, then $\llbracket M \rrbracket^{\lambda} a \Downarrow_{\pi}$.

vii) If $M \in \lambda$, and $M \Downarrow_{\beta}$, then $\llbracket M \rrbracket^{\lambda} a \Downarrow_{\pi}$.

Also the full abstraction result follows in exactly the same way as presented above for $\lambda\mu$. The equivalences '$\sim_{w\beta}$', '$\sim_{w\mathrm{H}}$', '$\sim_{w\mathrm{xH}}$', '$\sim_{\mathcal{A}_w}$', are defined for the λ-calculus by simply omitting the case for the context switch from the relevant definitions above, and we can show:

Theorem A.9 *For $M, N \in \lambda$, $\llbracket M \rrbracket^{\lambda} a \approx \llbracket N \rrbracket^{\lambda} a \iff M \sim_{w\mathrm{xH}} N \iff M \sim_{w\mathrm{H}} N \iff M \sim_{\mathcal{A}_w} N \iff M \sim_{w\beta} N$.*

which states that we have a fully-abstract semantics for the pure λ-calculus as well.