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

Steffen van Bakel and Maria Grazia Vigliotti

Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2BZ, UK
s.vanbakel@imperial.ac.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 research presented in this paper is part of an ongoing investigation into the suitability of classical logic in the context of programming languages with control. Rather than looking at how to encode known control features into calculi like the λ-calculus [24, 15], Parigot's $\lambda\mu$-calculus [47], or $\Lambda\mu$[1] [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 (perhaps better understood) π-calculus [43].

$\lambda\mu$ 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 Definition 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. 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 subterms 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 interpretations of $\lambda\mu$ into $\overline{\lambda}\mu\tilde{\mu}$ [26] and $\mathcal{X}$ (as introduced in [11]), respectively, respect equality, but do not respect reduction.

[1] The name $\Lambda\mu$ was first introduced in [52], 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.

Denotational semantics of $\lambda\mu$ has been studied by Streicher and Reus [54], 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 $D = C \rightarrow R$ 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 [54], 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. 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.

Here we define a semantics of $\lambda\mu$ using the π-calculus, which, with its dialects, over the last decades has proven to give an interesting and expressive model of computation. Encodings of variants of the pure λ-calculus started with Milner's [43] encoding of the lazy λ-calculus that led to more thorough investigations [44, 50, 21, 51, 12] (many more papers were written on the topic), also in the direction of object oriented calculi [35, 51]. The strength of the results that have been shown in those papers – like soundness, completeness, termination, and full abstraction – has led researchers to investigate interpretations into the π-calculus of calculi that have their foundation in classical logic [36, 10, 25, 17]. From these papers it might seem that the interpretation of such 'classical' calculi comes at a great expense; for example, to encode *typed* $\lambda\mu$, in [36] Honda, Yoshida, and Berger define an extension of Milner's encoding that uses a version of the π-calculus that is strongly typed; since reduction in $\mathcal{X}$ is not confluent, in [10], together with Cardelli the authors have shown preservation of reduction in $\mathcal{X}$ under the interpretation into the p-calculus with respect to the contextual ordering '$\sqsubseteq_{\mathrm{C}}$' (so not with respect to contextual equivalence '$\sim_{\mathrm{C}}$', nor weak bisimilarity '$\approx$'); in [25], Cimini, Sacerdoti Coen, and Sangiorgi define a non-compositional interpretation of $\overline{\lambda}\mu\tilde{\mu}$ that strongly depends on recursion, and does not regard the logical aspect at all.

We contribute to this line of research and 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 Definition 5.1) modulo contextual equivalence '$\sim_{\mathrm{C}}$'; here we restate those properties with respect to weak bisimilarity '$\approx$'. We show that by extending our output-based interpretation $\ulcorner M \lrcorner^{\mathrm{L}}\, a$ 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*, 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, 51] 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 '$\rightarrow_{\text{XH}}$' reductions through '$\rightarrow_{\pi}$' synchronisations directly, but modulo garbage collection, 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 Theorem 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 6.8), 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 6.4).

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 Theorem 7.1, we will 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 Theorem 7.6 we show that the encoding also respects equality on $\lambda\mathbf{x}$, but modulo weak bisimulation, and in Theorem 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 Theorem 8.4 we will show that to model lazy reduction on closed terms, this renaming is not needed.

We will show that all results obtained for $\lambda\mu$ can be shown for the λ-calculus as well; in Theorem 13.7 we will 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 terms that are equal under the semantics 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 $\mathit{0}$. 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 $\mathit{0}$. This then immediately implies that the interpretations of $\lambda y.\Delta\Delta$ and $\Delta\Delta$ *are not* bisimilar. However, these terms are both *unsolvable*, have both no head-normal form, and have the same set of approximants [56, 57] (*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 [50, 51] 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, considerably different technique: rather than reason through applicative bisimulation, we reason through weak approximation semantics that gets defined in this paper. First we characterise what is exactly the equivalence between terms in $\lambda\mu$ that is representable in the π-calculus through our encoding $\llbracket \cdot \rrbracket^{\mathrm{L}}\, \cdot$; as for Sangiorgi, this turns out to be *weak* equivalence (see Section 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 Theorem 7.1 we will show that our interpretation respects '$\rightarrow_{\mathrm{xH}}$' modulo '$\approx$', and in Theorem 7.6 that it even models '$=_{\mathrm{x}}$', the congruence generated by '$\rightarrow_{\mathrm{x}}$', from which a similar result for '$=_{\beta\mu}$' follows directly. In Theorem 12.1, we extend this result to *weak explicit head equivalence* '$\sim_{w\mathrm{xH}}$', the equivalence relation generated by '$\rightarrow_{\mathrm{xH}}$' 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 [56, 57], 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, 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 Section 1 and define a notion of *head-reduction* '$\rightarrow_{\mathrm{H}}$'. In Section 2 we revisit the π-calculus, enriched with *pairing*, and will discuss some of the context and background of our work in Section 3. In Section 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 Section 5, and in Section 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 Section 7. In Section 8 we will show that the operation of renaming we have defined in Section 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 $\ulcorner M \lrcorner^{\mathrm{L}} a \approx \ulcorner N \lrcorner^{\mathrm{L}} a$ if and only if $M \sim_{w\beta\mu} N$, in Section 9 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 '$\sim_{w\mathrm{H}}$' and '$\sim_{w\mathrm{XH}}$', 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 Section 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 '$\sim_{w\mathrm{H}}$' and '$\sim_{w\beta\mu}$'. Then, in Section 12, we will show that our logical interpretation is fully abstract with respect to weak bisimilarity '$\approx$' on processes and the equivalences '$\sim_{w\mathrm{XH}}$', '$\sim_{w\mathrm{H}}$', '$\sim_{\mathcal{A}_w}$', and '$\sim_{w\beta\mu}$' on pure $\lambda\mu$-terms.

To conclude, in Section 13, 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 (Theorem 3.3) in Corollary 13.5, 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 $\overrightarrow{\langle \alpha_i := N_i \cdot \beta_i \,\rangle}$ 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', since the latter represents a logical implication. By abuse of notation, we will use '$\in$' also as an abbreviation of 'occurs in', for example for sequences or terms.

1 The $\lambda\mu$ calculus

In this section, we will briefly discuss Parigot's $\lambda\mu$-calculus [47]; 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 Section 4 we will define explicit head-reduction for $\lambda\mu\mathbf{x}$, a variant of $\lambda\mu$ with explicit substitution *à la* Bloo and Rose's $\lambda\mathbf{x}$ [20], and will show full abstraction results for $\lambda\mu\mathbf{x}$ later in the paper; 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 the concept of *named* terms, and adding the functionality of a *context switch*, allowing arguments to be fed to subterms.

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:

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

We will occasionally write *Cmd* 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 $\mu\alpha.\mathit{Cmd}$ binds α in *Cmd*, and the notions of free variables $\mathit{fv}(M)$ and names $\mathit{fn}(M)$ 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.

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 $\lambda\mu$, reduction of terms is expressed via implicit substitution; as usual, $M\{N/x\}$ stands for the (term) 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 $[\alpha]P$ is replaced by $[\gamma]PN$.[2] For reasons of clarity, and because below we will present a version of $\lambda\mu$

[2] This notion is often defined as $M\{N/\alpha\}$, where every (pseudo) sub-term of the form $[\alpha]P$ is replaced by $[\alpha]PN$; 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 (*S-sub*) in

that makes the substitution explicit, we define the structural substitution formally.

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:

$$\begin{array}{rcll}
([\alpha]M)\,\{N{\cdot}\gamma/\alpha\} & \triangleq & [\gamma](M\{N{\cdot}\gamma/\alpha\})N & \\
([\beta]M)\,\{N{\cdot}\gamma/\alpha\} & \triangleq & [\beta](M\{N{\cdot}\gamma/\alpha\}) & (\alpha \neq \beta)\\
(\mu\beta.Cmd)\,\{N{\cdot}\gamma/\alpha\} & \triangleq & \mu\beta.Cmd\{N{\cdot}\gamma/\alpha\} & \\
x\,\{N{\cdot}\gamma/\alpha\} & \triangleq & x & \\
(\lambda x.M)\,\{N{\cdot}\gamma/\alpha\} & \triangleq & \lambda x.M\{N{\cdot}\gamma/\alpha\} & \\
(PQ)\,\{N{\cdot}\gamma/\alpha\} & \triangleq & P\{N{\cdot}\gamma/\alpha\}\;Q\{N{\cdot}\gamma/\alpha\} &
\end{array}$$

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*

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

as well as the *simplification rules*

$$\begin{array}{rrcll}
\textit{renaming}: & [\beta]\mu\gamma.Cmd & \rightarrow & Cmd\{\beta/\gamma\} & \\
\textit{erasing}: & \mu\alpha.[\alpha]M & \rightarrow & M & (\alpha \notin fn(M))
\end{array}$$

which are added mainly to simplify the presentation of results.

ii) We use the contextual rules:[3]

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

iii) We use '$\rightarrow^{*}_{\beta\mu}$' for the pre-congruence[4] based on these rules, '$=_{\beta\mu}$' for the congruence, write $M \rightarrow^{nf}_{\beta\mu} N$ if $M \rightarrow^{*}_{\beta\mu} N$ and N is in normal form, $M \rightarrow^{hnf}_{\beta\mu} N$ if $M \rightarrow^{*}_{\beta\mu} N$ and N is in head-normal form, $M{\Downarrow}$ if there exists a finite reduction path starting from M,[5] 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 [49]; so we have:

Proposition 1.5 ([49]) *If $M =_{\beta\mu} N$ and $M \rightarrow^{*}_{\beta\mu} P$, then there exists Q such that $P \rightarrow^{*}_{\beta\mu} Q$ and $N \rightarrow^{*}_{\beta\mu} Q$.*

The intuition behind the structural rule is given by [32]: '*in a $\lambda\mu$-term $\mu\alpha.M$ of type $A{\rightarrow}B$, only the subterms named by α are* really *of type $A{\rightarrow}B$ (...); 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

Definition 4.3. Moreover, when making this substitution explicit, bound and free occurrences of the same name would be introduced, violating Barendregt's convention.

[3] 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.

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

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

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

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

think of $[\alpha]M$ as storing the type of M amongst the alternative conclusions by naming it α.

Parigot showed in [48] that typeable terms are strongly normalisable. That paper also defines the extensional rules

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

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, Figure 1 shows that it is possible to inhabit Peirce's Law (due to [46]). The underlying logic of the system of Definition 1.2 corresponds to *minimal classical logic* [5].

We also consider the notion of *head reduction*; Wadsworth [56] defined that 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

$$\lambda x_1x_2\cdots x_n.((\lambda y.M)N)L_1L_2\cdots L_m \quad (n \geq 0, m \geq 0)$$

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

$$\lambda x_1x_2\cdots x_n.yL_1L_2\cdots L_m \quad (n \geq 0, m \geq 0)$$

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* '$\to_{\text{H}}$' 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:

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

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

The following is straightforward:

Proposition 1.8 (‘$\rightarrow_{\text{H}}$’ IMPLEMENTS $\lambda\mu$’S HEAD REDUCTION) *If $M \rightarrow^{*}_{\beta\mu} N$ with N in* HNF *(so $M \rightarrow^{hnf}_{\beta\mu} N$), then there exists $\boldsymbol{H}$ such that $M \rightarrow^{nf}_{\text{H}} \boldsymbol{H}$ (so $\boldsymbol{H}$ is in $\rightarrow_{\text{H}}$-normal form) and $\boldsymbol{H} \rightarrow^{*}_{\beta\mu} N$ without using ‘$\rightarrow_{\text{H}}$’.*

Notice that $\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)) \rightarrow_{\text{H}} \lambda f.f((\lambda x.f(xx))(\lambda x.f(xx)))$ and that this last term is in HNF, and in $\rightarrow_{\text{H}}$-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 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 by Abadi and Gordon [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 that we want to communicate the interface of functions, through simultaneously transmitting the names of the input and of the output channel of the process that represents the function. 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. We could consider the standard π-calculus; however, the details of the interpretation would change (more replication would be needed).

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

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

ii) Processes are defined by:

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

iii) We see, as usual, ν as a binder, and call the name n *bound* in $(\nu n)\,\mathsf{P}$, x bound in $a(x).\mathsf{P}$ and x,y bound in *let* $\langle x,y \rangle = \mathsf{p}$ *in* P; we write $bn\,(\mathsf{P})$ 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 $fn(\mathsf{P})$ for the set of free names in P. We accept the normal convention on the separation of free and bound names, using α-conversion when necessary. We call a in $(\nu a)\,\mathsf{P}$ a *hidden* channel.

iv) A *context* $\mathsf{C}[\cdot]$ is a process with a single (process) hole, and we write $\mathsf{C}[\mathsf{P}]$ when filling the hole with P.

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

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

vii) As in [51], we write $a \rightarrow \overline{b}$ for the *forwarder* $a(x).\overline{b}\,x$, and $\overline{x}(w).\mathsf{P}$ for $(\nu w)\,(\overline{x}\,w.\mathsf{P})$.

Notice that the pairing in data is *not* recursive. Data occurs only in two cases: $\overline{a}\,\mathsf{p}$ and *let* $\langle x,y \rangle = \mathsf{p}$ *in* P, and then p is either a single name, or a pair of names; we therefore do not allow $a(\langle x,y \rangle).\mathsf{P}$, nor $\overline{a}\,\langle \langle b,c \rangle,d \rangle.\mathsf{P}$, nor $\overline{\langle b,c \rangle}\,\mathsf{p}.\mathsf{P}$, nor $(\nu \langle a,b \rangle)\,\mathsf{P}$, nor *let* $\langle \langle a,b \rangle,y \rangle = p$ *in* P, etc. So substitution $\mathsf{P}\{\mathsf{p}/x\}$ is a partial operation, which depends on the places in P where x occurs; whenever we use $\mathsf{P}\{\mathsf{p}/x\}$, 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:

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

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 $P\{a/x,b/y\}$. 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:

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

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

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

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 $P \to_\pi^* Q$ 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 $P \to_\pi^* Q$ 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* '$\dot{\approx}$' is the largest symmetric relation such that $P \mathrel{\dot{\approx}} Q$ 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 $P \to_\pi^* P'$, then there exists Q' such that $Q \to_\pi^* Q'$ and $P' \mathrel{\dot{\approx}} Q'$;

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

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

Lemma 2.5 (Private resources lemma (*cf.* [44, 51])) *Let $x \neq c$ at most only be used as output channel in P and Q, and not appear in R, then*

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

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

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

Proof: All parts follow easily. Part 1, 4, and 5 are shown in [44, 51] (see Theorem 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, Q not contain a and a not in p, then*

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

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 it. 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 '$\rightarrow_{\mathrm{L}}$' [3] to normal form up to substitution (Theorem 3.3); this was later generalised to β-equality, but using weak bisimilarity [51].

It is defined by:

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

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

Milner calls $\ulcorner x := M \lrcorner^{\mathrm{M}}$ an *environment entry*; it corresponds to a *closure* in Krivine's machine [38], that are also grouped in an environment; it could be omitted from the definition above, but is of use separately.

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

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

Now reduction can continue in (the encoding of) M, but not in N that is still guarded by the input on x, which will not be used until the evaluation of $\ulcorner M \lrcorner^{\mathrm{M}}\, a$ 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 Theorem 13.4.

Notice that, in $\ulcorner MN \lrcorner^{\mathrm{M}}\, a$, 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*, '$\rightarrow_{\mathrm{L}}$'.

Milner states an Operational Soundness result for his interpretation:

Theorem 3.3 ([43]) *For closed λ-term M, either $M\Uparrow$ and $\ulcorner M \lrcorner^{\mathrm{M}}\, u \Uparrow_{\pi}$, or $M \rightarrow_{\mathrm{L}}^{*} \lambda y.R\,\overrightarrow{\{N/x\}}$, and*

$$
\ulcorner M \lrcorner^{\mathrm{M}}\, u \;\rightarrow_{\pi}^{*}\; (\overrightarrow{\nu x}\,)\,(\ulcorner \lambda y.R \lrcorner^{\mathrm{M}}\, u \mid \overrightarrow{\ulcorner x := N \lrcorner^{\mathrm{M}}}\,).
$$

Although obviously the intention in [43] is that the substitutions $\{\overrightarrow{N/x}\,\}$ in Theorem 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. We will come back to this in Theorem 13.4, where we restate Milner's result, but formulated with explicit substitution.

Moreover, notice that $(\lambda x.xx)(\lambda y.y) \rightarrow_{\mathrm{L}}^{*} \lambda y.y$: however, we need garbage collection to run $\ulcorner (\lambda x.xx)(\lambda y.y) \lrcorner^{\mathrm{M}}\, a$ to $\ulcorner \lambda y.y \lrcorner^{\mathrm{M}}\, a$, which Milner does not consider; without garbage collection, it runs to

$$
(\nu z)\,((\nu z_1)\,(\ulcorner \lambda y.y \lrcorner^{\mathrm{M}}\, a \mid \ulcorner z_1 := z \lrcorner^{\mathrm{M}}) \mid \ulcorner z := \lambda y.y \lrcorner^{\mathrm{M}}),
$$

so also in this case the formulation of Milner's result is imprecise. This anomaly was addressed in [44] where $\ulcorner(\lambda x.M)N\lrcorner \approx \ulcorner M\{N/x\}\lrcorner$ is stated using weak bisimilarity, and in [51] using ground bisimilarity, as we will discuss below.

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 [51] 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; [55] 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

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

provided that $|\vec{x}\,| = |\vec{y}\,|$. The new version of the encoding now becomes:

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

Milner shows that

Lemma 3.4 $(\nu x)\,(\ulcorner M\lrcorner^{\mathrm{P}} \mid {!\,x}.\ulcorner N\lrcorner^{\mathrm{P}}\,) \approx \ulcorner M\{N/x\}\lrcorner^{\mathrm{P}}$

This result is stated using full weak bisimilarity. 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) (see Definition 6.7).

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

Theorem 3.5 (REPLICATION THEOREM [44, 51]) *If x occurs in P, Q, and R only in output subject position (as subjects of output prefixes, as negative subjects), then*

$$\begin{array}{rcll}
(\nu x)\,(P \mid Q \mid {!\,x(z)}.R) & \approx & (\nu x)\,(P \mid {!\,x(z)}.R) \mid (\nu x)\,(Q \mid {!\,x(z)}.R) & (12)\\
(\nu x)\,({!\,P} \mid {!\,x(z)}.R) & \approx & {!\,(\nu x)}\,(P \mid {!\,x(z)}.R) & (13)
\end{array}$$

and[6] *if x does not occur in π, then*

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

This permits the '$\approx$' steps in

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

$$
\begin{array}{rcl}
\multicolumn{3}{l}{(\nu x)\,(\ulcorner M_1M_2 \rrbracket^{\mathrm{P}}\,u \mid !\,x.\ulcorner N \rrbracket^{\mathrm{P}}\,)} \\
& \triangleq & (\nu x)\,((\lambda u)\,(\nu v)\,(\ulcorner M_1 \rrbracket^{\mathrm{P}}\,v \mid (\nu z)\,(\overline{v}.[zu] \mid !\,z.\ulcorner M_2 \rrbracket^{\mathrm{P}}\,))u \mid !\,x.\ulcorner N \rrbracket^{\mathrm{P}}\,) \\
& = & (\nu x)\,((\nu v)\,(\ulcorner M_1 \rrbracket^{\mathrm{P}}\,v \mid (\nu z)\,(\overline{v}.[zu] \mid !\,z.\ulcorner M_2 \rrbracket^{\mathrm{P}}\,)) \mid !\,x.\ulcorner N \rrbracket^{\mathrm{P}}\,) \\
(12) & \approx & (\nu v)\,((\nu x)\,(\ulcorner M_1 \rrbracket^{\mathrm{P}}\,v \mid !\,x.\ulcorner N \rrbracket^{\mathrm{P}}\,) \mid (\nu x)\,((\nu z)\,(\overline{v}.[zu] \mid !\,z.\ulcorner M_2 \rrbracket^{\mathrm{P}}\,) \mid !\,x.\ulcorner N \rrbracket^{\mathrm{P}}\,)) \\
& \equiv & (\nu v)\,((\nu x)\,(\ulcorner M_1 \rrbracket^{\mathrm{P}}\,v \mid !\,x.\ulcorner N \rrbracket^{\mathrm{P}}\,) \mid (\nu z)\,(\overline{v}.[zu] \mid (\nu x)\,(!\,z.\ulcorner M_2 \rrbracket^{\mathrm{P}} \mid !\,x.\ulcorner N \rrbracket^{\mathrm{P}}\,))) \\
(13) & \approx & (\nu v)\,((\nu x)\,(\ulcorner M_1 \rrbracket^{\mathrm{P}}\,v \mid !\,x.\ulcorner N \rrbracket^{\mathrm{P}}\,) \mid (\nu z)\,(\overline{v}.[zu] \mid !\,(\nu x)\,(z.\ulcorner M_2 \rrbracket^{\mathrm{P}} \mid !\,x.\ulcorner N \rrbracket^{\mathrm{P}}\,))) \\
(14) & \approx & (\nu v)\,((\nu x)\,(\ulcorner M_1 \rrbracket^{\mathrm{P}}\,v \mid !\,x.\ulcorner N \rrbracket^{\mathrm{P}}\,) \mid (\nu z)\,(\overline{v}.[zu] \mid !\,z.(\nu x)\,(\ulcorner M_2 \rrbracket^{\mathrm{P}} \mid !\,x.\ulcorner N \rrbracket^{\mathrm{P}}\,))) \\
(IH) & \approx & (\lambda u)\,(\nu v)\,(\ulcorner M_1\{N/x\} \rrbracket^{\mathrm{P}}\,v \mid (\nu z)\,(\overline{v}.[zu] \mid !\,z.\ulcorner M_2\{N/x\} \rrbracket^{\mathrm{P}}\,))\,u \\
& \triangleq & \ulcorner M_1\{N/x\}M_2\{N/x\} \rrbracket^{\mathrm{P}}\,u
\end{array}
$$

Notice that, the notational differences notwithstanding, the encoding $\ulcorner \cdot \rrbracket^{\mathrm{P}}$ is input-based, and a direct rewrite of the original one.

After Milner's encodings, many variants followed; for example, [51] 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 $\llbracket \cdot \rrbracket^{\mathcal{N}}$ [7] of the lazy λ-calculus is defined through:

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

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, [51] shows a number of results; first it shows:

$$
\llbracket (\lambda x.M)\,N \rrbracket^{\mathcal{N}} \;\overset{(1)}{\xrightarrow{\tau}{}^{2}_{d}}\; (\nu x)\,(\llbracket M \rrbracket^{\mathcal{N}} \mid !\,x(w).\llbracket N \rrbracket^{\mathcal{N}}\,) \;\overset{(2)}{\approx_g}\; \llbracket M\{N/x\} \rrbracket^{\mathcal{N}}
$$

(where '$\xrightarrow{\tau}_d$' is the deterministic (silent) transition and '$\approx_g$' is ground bisimilarity) which leads to:[8]

$$
\begin{array}{rcl}
M \rightarrow_{\mathrm{L}} N & \overset{(3)}{\Rightarrow} & \llbracket M \rrbracket^{\mathcal{N}} \xrightarrow{\tau}{}^{2}_{d}\approx_g \llbracket N \rrbracket^{\mathcal{N}} \\
M =_\beta N & \overset{(4)}{\Rightarrow} & \llbracket M \rrbracket^{\mathcal{N}} \approx_g \llbracket N \rrbracket^{\mathcal{N}}
\end{array}
$$

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 Theorem 7.1 and Theorem 7.7 below.

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

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

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.

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

[8] In [51], 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.

This result comes with caveats, however: as shown by Ong [45], 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 communication.

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 [50] 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 [50], presented almost all without proof, are shown for closed terms only,[9] 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:

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

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

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

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

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

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 occurrence at the time*, rather than all together in one fell swoop. (We come back to those results in Section 13.) 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.

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 $\lambda\mu$, $\overline{\lambda}\mu\tilde{\mu}$, or $\mathcal{X}$.

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

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*.[10] The syntax of processes considered there is

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

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:

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

as the basic synchronisation rule, as well as

$$\begin{array}{rcl} \mathsf{C}[(\nu\vec{a}\,)\,(\overline{x}\,\vec{a}\ |\,P)]\,|\,!\,x(\vec{y}\,).Q & \searrow_r & \mathsf{C}[(\nu\vec{a}\,)\,(P\{\overrightarrow{a/y}\,\}\,|\,Q)]\,|\,!\,x(\vec{y}\,).Q \\ (\nu x)\,(!\,x(\vec{y}\,).Q) & \searrow_g & 0 \end{array}$$

where $\mathsf{C}[\cdot]$ 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, Bellin and Scott [19] give 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 statement here, we observe that [19] uses the standard syntax for the polyadic π-calculus

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

similar to the one we use here (see Definition 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:

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

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:

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

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

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 to add the *let*-construct with use of pairs of names as we have introduced in this paper in Definition 2.1.

In [10] an interpretation into π of the sequent calculus $\mathcal{X}$, which enjoys the Curry-Howard isomorphism for Gentzen's LK, is defined and shown to respect reduction. It is formulated as *'if* $P \rightarrow_{\mathcal{X}} Q$*, then* $[\![P]\!] \,{}_{\mathrm{c}}\!\sqsupseteq [\![Q]\!]$', allowing $[\![P]\!]$ to have more observable behaviour than $[\![Q]\!]$. The main reason for this is that reduction in $\mathcal{X}$ is non-confluent; taking $P\,\widehat{\alpha} \dagger \widehat{x}\,Q$, with α not in P and x not in Q, then $P\,\widehat{\alpha} \dagger \widehat{x}\,Q \rightarrow_{\mathcal{X}} P$ and $P\,\widehat{\alpha} \dagger \widehat{x}\,Q \rightarrow_{\mathcal{X}} Q$, where subterms are removed during reduction. Simulation of this in the π-calculus through $[\![P\,\widehat{\alpha} \dagger \widehat{x}\,Q]\!] = [\![P]\!] \mid [\![Q]\!]$ creates a process that can simulate both the reductions in P and Q; but since the π-calculus has no feature to erase part of a process, $[\![P\,\widehat{\alpha} \dagger \widehat{x}\,Q]\!]$ does not run to either $[\![P]\!]$ or $[\![Q]\!]$. All possible normal forms of a term P are represented, in parallel, in $[\![P]\!]$ and it is not guaranteed that either of them can be considered garbage. In that light, it is impossible to show *'if* $P \rightarrow_{\mathcal{X}} Q$*, then* $[\![P]\!] \approx [\![Q]\!]$' for any interpretation of $\mathcal{X}$ into the π-calculus; as argued in [10], this is natural in the context of non-confluent, symmetric sequent calculi.

An interpretation of $\overline{\lambda}\mu\tilde{\mu}$ 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* [20] for the λ-calculus, 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 Theorem 13.7), 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.

Definition 4.1 ($\lambda\mu\mathbf{x}$) *i*) The syntax of the $\lambda\mu$ *calculus with explicit substitution*, $\lambda\mu\mathbf{x}$, is defined by:

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

where x ranges over an infinite countable set of *variables*, and a and β range over an infinite countable set of *names*. *Bound variables and names* of terms are defined by:

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

and we call a variable or name *free* in M (using $fv(\cdot)$ and $fn(\cdot)$, respectively) 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.[11]

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

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

$$\begin{array}{llcll}
\textit{Main reduction rules}: & (\lambda x.M)\,N & \rightarrow & M\,\langle x := N\rangle & \\
& (\mu\alpha.Cmd)\,N & \rightarrow & \mu\gamma.(Cmd\,\langle \alpha := N\cdot\gamma\rangle) & (\gamma\,\textit{fresh}) \\
& \mu\beta.[\beta]\,M & \rightarrow & M & (\beta \notin \mathit{fn}(M)) \\
& [\beta](\mu\gamma.Cmd) & \rightarrow & Cmd\{\beta/\gamma\}\ ^{12} & \\
\textit{Term substitution rules}: & x\,\langle x := N\rangle & \rightarrow & N & \\
& M\,\langle x := N\rangle & \rightarrow & M & (x \notin \mathit{fv}(M)) \\
& (\lambda y.M)\,\langle x := N\rangle & \rightarrow & \lambda y.(M\,\langle x := N\rangle) & \\
& (PQ)\,\langle x := N\rangle & \rightarrow & (P\,\langle x := N\rangle)(Q\,\langle x := N\rangle) & \\
& (\mu\alpha.[\beta]\,M)\,\langle x := N\rangle & \rightarrow & \mu\alpha.[\beta]\,M\,\langle x := N\rangle & \\
\textit{Structural substitution rules}: & M\,\langle \alpha := N\cdot\gamma\rangle & \rightarrow & M & (\alpha \notin \mathit{fn}(M)) \\
& (\lambda x.M)\,\langle \alpha := N\cdot\gamma\rangle & \rightarrow & \lambda x.M\,\langle \alpha := N\cdot\gamma\rangle & \\
& (PQ)\,\langle \alpha := N\cdot\gamma\rangle & \rightarrow & (P\,\langle \alpha := N\cdot\gamma\rangle)\,(Q\,\langle \alpha := N\cdot\gamma\rangle) & \\
& ([\alpha]M)\,\langle \alpha := N\cdot\gamma\rangle & \rightarrow & [\gamma]M\,\langle \alpha := N\cdot\gamma\rangle\,N & \\
& ([\beta]M)\,\langle \alpha := N\cdot\gamma\rangle & \rightarrow & [\beta]M\,\langle \alpha := N\cdot\gamma\rangle & (\alpha \neq \beta) \\
& (\mu\delta.Cmd)\,\langle \alpha := N\cdot\gamma\rangle & \rightarrow & \mu\delta.Cmd\,\langle \alpha := N\cdot\gamma\rangle &
\end{array}$$

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

iv) We use '$\rightarrow_{:=}$' 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 '$\rightarrow_{\mathrm{x}}$'.

v) To ease notation, we will use 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; each entry in S concerns a unique variable or name. We write $x \in S$ if $\langle x := N\rangle \in S$ and say that S is *defined on* x, and write $S\backslash x$ for S_1S_2 if $S = S_1\langle x := N\rangle S_2$ and similarly for $\alpha \in S$ and $S\backslash\alpha$. We write S_c if S is *only* defined on c, *i.e.* $S_c = \langle c := N\rangle$ or $S_c = \langle c := N\cdot\gamma\rangle$.

Notice that 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.

We do not add rules like

[11] Note that here, for the explicit case, the convention to 'reuse' α rather than introduce the new name γ, would have us write $M\,\langle \alpha := N\cdot\alpha\rangle$ which would create a violation of Barendregt's convention since in $M\,\langle \alpha := N\cdot\beta\rangle$, α is bound and β is free.

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

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

since as in [20], this would introduce undesired non-termination.

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:

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

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

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 we have:

Proposition 4.2 ($\lambda\mu\mathbf{x}$ IMPLEMENTS $\lambda\mu$-REDUCTION) *$M \rightarrow_{\beta\mu} N \Rightarrow M \rightarrow^*_{\mathrm{x}} N$.*

Proof: Straightforward. □

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

Definition 4.3 (TYPE ASSIGNMENT FOR $\lambda\mu\mathbf{x}$) Using the notion of types in Definition 1.2, type assignment for $\lambda\mu\mathbf{x}$ is defined by:

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

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

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

We write $\Gamma \vdash_{\beta\mu\mathrm{x}} M : A \mid \Delta$ 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 4.4 (WEAKENING AND THINNING) *If $\Gamma \vdash_{\beta\mu\mathrm{x}} M : A \mid \Delta$ and either $\Gamma' = \{\, x{:}B \in \Gamma \mid x \in fv(M)\,\}$ and $\Delta' = \{\, \alpha{:}B \in \Delta \mid \alpha \in fn(M)\,\}$ (thinning), or $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$ (weakening), then $\Gamma' \vdash_{\beta\mu\mathrm{x}} M : A \mid \Delta'$.*

Proof: Straightforward. □

We can now show:

Theorem 4.5 (SUBJECT REDUCTION) *If $P \rightarrow_{\mathrm{x}} Q$, and $\Gamma \vdash_{\beta\mu\mathrm{x}} P : A$ then $\Gamma \vdash_{\beta\mu\mathrm{x}} Q : A$.*

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

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

$$\dfrac{\dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ \Gamma,x{:}A \vdash M : B \mid \Delta\end{array}}{\Gamma \vdash \lambda x.M : A{\to}B \mid \Delta}\,(\to I) \qquad \begin{array}{c}\boxed{\qquad\qquad}\\ \Gamma \vdash N : A \mid \Delta\end{array}}{\Gamma \vdash (\lambda x.M)\,N : B \mid \Delta}\,(\to E) \qquad \dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ \Gamma,x{:}A \vdash M : B \mid \Delta\end{array} \qquad \begin{array}{c}\boxed{\qquad\qquad}\\ \Gamma \vdash N : A \mid \Delta\end{array}}{\Gamma \vdash M\,\langle x := N\rangle : B \mid \Delta}\,(T\text{-}sub)$$

$(\mu\alpha.Cmd)N \to \mu\gamma.Cmd\,\langle\alpha := N{\cdot}\gamma\rangle$: We have two cases:
/SsubThen $\mu\gamma.([\alpha]M)\,\langle\alpha := N{\cdot}\gamma\rangle = \mu\gamma.[\gamma]\,M\,\langle\alpha := N{\cdot}\gamma\rangle N$. Then the derivation is shaped like

$$Cmd = [\alpha]M: \quad \dfrac{\dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ \Gamma \vdash M : B{\to}A \mid \alpha{:}B{\to}A,\Delta\end{array}}{\Gamma \vdash \mu\alpha.[\alpha]\,M : B{\to}A \mid \Delta}\,(\mu) \qquad \begin{array}{c}\boxed{\qquad\qquad}\\ \Gamma \vdash N : B \mid \Delta\end{array}}{\Gamma \vdash (\mu\alpha.[\alpha]\,M)\,N : A \mid \Delta}\,(\to E)$$

from which we can construct

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ \Gamma \vdash M : B{\to}A \mid \alpha{:}B{\to}A,\Delta\end{array} \qquad \begin{array}{c}\boxed{\qquad\qquad}\\ \Gamma \vdash N : B \mid \Delta\end{array}}{\Gamma \vdash M\,\langle\alpha := N{\cdot}\gamma\rangle : B{\to}A \mid \gamma{:}A,\Delta}\,(S\text{-}sub) \qquad \dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ \Gamma \vdash N : B \mid \Delta\end{array}}{\Gamma \vdash N : B \mid \gamma{:}A,\Delta}\,(Weak)}{\Gamma \vdash (M\,\langle\alpha := N{\cdot}\gamma\rangle)N : A \mid \gamma{:}A,\Delta}\,(\to E)}{\Gamma \vdash \mu\gamma.[\gamma]\,M\,\langle\alpha := N{\cdot}\gamma\rangle N : A \mid \Delta}\,(\mu)$$

Cmd $= [\beta]M$, *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

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

from which we can construct:

$$\dfrac{\dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ \Gamma \vdash_{\beta\mu x} M : C \mid \alpha{:}A{\to}B,\beta{:}C,\Delta\end{array} \qquad \begin{array}{c}\boxed{\qquad\qquad}\\ \Gamma \vdash_{\beta\mu x} N : A \mid \beta{:}C,\Delta\end{array}}{\Gamma \vdash_{\beta\mu x} M\,\langle\alpha := N{\cdot}\gamma\rangle : C \mid \gamma{:}B,\beta{:}C,\Delta}\,(S\text{-}sub)}{\Gamma \vdash_{\beta\mu x} \mu\gamma.[\beta]\,M\,\langle\alpha := N{\cdot}\gamma\rangle : B \mid \beta{:}C,\Delta}\,(\mu)$$

$\mu\delta.[\beta]\,\mu\gamma.[\alpha]\,M \to \mu\delta.[\alpha]\,M\{\beta/\gamma\}$: (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, since β and γ have the same type.

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

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

$$\dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ \Gamma \vdash M : A \mid \alpha{:}C,\Delta\end{array}}{\Gamma \vdash \mu\alpha.[\alpha]\,M : A \mid \Delta}\,(\mu)$$

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

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

$$\dfrac{\begin{array}{c}\boxed{\phantom{\Gamma,x{:}B \vdash M : A \mid \Delta}}\\ \Gamma,x{:}B \vdash M : A \mid \Delta\end{array} \qquad \begin{array}{c}\boxed{}\\ \Gamma \vdash N : B \mid \Delta\end{array}}{\Gamma \vdash M\langle x := N\rangle : A \mid \Delta}\;(\mathit{T\text{-}sub})$$

Since $x \notin fv(M)$, by thinning from the left-hand sub-derivation also $\Gamma \vdash_{\beta\mu\mathrm{x}} M : A \mid \Delta$.
$(\mu\beta.[\alpha]\,M)\,\langle\alpha := N\cdot\gamma\rangle \to \mu\beta.[\gamma]\,M\,\langle\alpha := N\cdot\gamma\rangle N$: Then the derivation is shaped like

$$\dfrac{\dfrac{\begin{array}{c}\boxed{\phantom{\Gamma \vdash M : B\to C \mid \alpha{:}B\to C,\beta{:}A,\Delta}}\\ \Gamma \vdash M : B\to C \mid \alpha{:}B\to C,\beta{:}A,\Delta\end{array}}{\Gamma \vdash \mu\beta.[\alpha]\,M : A \mid \alpha{:}B\to C,\Delta}\;(\mu) \qquad \begin{array}{c}\boxed{}\\ \Gamma \vdash N : B \mid \Delta\end{array}}{\Gamma \vdash (\mu\beta.[\alpha]M)\,\langle\alpha := N\cdot\gamma\rangle : A \mid \gamma{:}C,\Delta}\;(\mathit{S\text{-}sub})$$

from which we can construct:

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

□

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. 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 main reduction rules are as before:

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

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

$$\begin{array}{rrcll}
(hv): & xS_0M_1S_1\cdots M_nS_n & \to & NS_0M_1S_1\cdots M_nS_n & (n \geq 0,\ \langle x := N\rangle \in S_n) \\
(\lambda S): & (\lambda y.M)S & \to & \lambda y.(MS) & \\
(hn): & (\mu\delta.[\alpha]M)S & \to & (\mu\delta.[\gamma]M\langle\alpha := N\cdot\gamma\rangle N)S\backslash\alpha & (\langle\alpha := N\cdot\gamma\rangle \in S,\ M \neq \mu\beta.\mathit{Cmd}) \\
(nS): & (\mu\delta.[\alpha]M)S & \to & \mu\delta.[\alpha]MS & (\alpha \notin S,\ M \neq \mu\beta.\mathit{Cmd}) \\
(gc): & MS & \to & MS\backslash c & (c \in S, c \notin M)
\end{array}$$

where in (hv) each S_i can be empty, except for S_n.

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

$$\begin{array}{ll}
(\mu\alpha.[\alpha]\lambda y.y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\,\lambda x.x))(\lambda z.z) & \rightarrow_{\mathrm{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 & \rightarrow_{\mathrm{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) & \rightarrow_{\mathrm{XH}}(\lambda 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) & \rightarrow_{\mathrm{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 & \rightarrow_{\mathrm{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 & \rightarrow_{\mathrm{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 & \rightarrow_{\mathrm{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 & \rightarrow_{\mathrm{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 & \rightarrow_{\mathrm{XH}}(\mu) \\
\mu\gamma.[\gamma]\,\mu\gamma'.([\alpha]\lambda x.x)\,\langle\delta:=\mu\delta'.[\alpha]\lambda x.x\cdot\gamma'\rangle\,\langle\alpha:=\lambda z.z\cdot\gamma\rangle & \rightarrow_{\mathrm{XH}}(gc) \\
\mu\gamma.[\gamma]\,\mu\gamma'.[\alpha]\lambda x.x\,\langle\alpha:=\lambda z.z\cdot\gamma\rangle & \rightarrow_{\mathrm{XH}}(hn) \\
\mu\gamma.[\gamma]\,\mu\gamma'.[\gamma]\,\lambda x.x\,\langle\alpha:=\lambda z.z\cdot\gamma\rangle\,(\lambda z.z) & \rightarrow_{\mathrm{XH}}(gc) \\
\mu\gamma.[\gamma]\;\mu\gamma'.[\gamma]\;(\lambda x.x)(\lambda z.z) & \rightarrow_{\mathrm{XH}}(R) \\
\mu\gamma.[\gamma]\;\;(\lambda x.x)(\lambda z.z)\;\rightarrow_{\mathrm{XH}}(C)\;(\lambda x.x)(\lambda z.z)\;\rightarrow_{\mathrm{XH}}(\beta)\;x\,\langle x:=\lambda z.z\rangle & \rightarrow_{\mathrm{XH}}(hv)\quad\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 '$\rightarrow_{\mathrm{XH}}$'.

$$M \rightarrow N \;\;\Rightarrow\;\; \begin{cases} \lambda x.M & \rightarrow\;\; \lambda x.N \\ ML & \rightarrow\;\; NL \\ \mu\alpha.[\beta]\,M & \rightarrow\;\; \mu\alpha.[\beta]\,N \quad (\alpha\neq\beta \vee \alpha\in M,\; M\neq\mu\delta.Cmd) \\ MS & \rightarrow\;\; NS \end{cases}$$

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 μ-redexes in $(\mu\alpha.[\beta]P)\vec{Q}$ have been contracted.

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:

$$(\lambda y.M)\,\langle x:=N\rangle \;\rightarrow\; \lambda y.(M\,\langle x:=N\rangle)\quad (x=hv(M))$$

but that would imply that a reduction like

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

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:

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

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

Definition 5.2 The normal forms with respect to '$\rightarrow_{\mathrm{XH}}$' are defined through the grammar:

$$\begin{array}{rcll}
\mathbf{N} & ::= & \lambda x.\mathbf{N} & \\
& \mid & xM_1S_1\cdots M_nS_n & (n\geq 0, x\not\in S_i) \\
& \mid & \mu\alpha.[\beta]\,\mathbf{N} & (\alpha\neq\beta\vee\alpha\in\mathbf{N},\;\mathbf{N}\neq\mu\gamma.[\delta]\,\mathbf{N}')
\end{array}$$

It is straightforward to check that these terms are indeed the normal forms with respect to '$\rightarrow_{\mathrm{XH}}$'.

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

Lemma 5.3 i) $M\,\langle x:=N\rangle \rightarrow^{*}_{:=} M\{N/x\}$.

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

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

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

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

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

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 $P \equiv xS_1M_1\cdots S_nM_nS_{n+1}$ with $\langle x := N\rangle \in S_{n+1}$, and $Q \equiv NS_1M_1\cdots S_nM_nS_{n+1}$. Let $xS_1M_1\cdots S_nM_nS_{n+1} \rightarrow^{nf}_{:=} NM'_1\cdots M'_n \equiv R \equiv S$. Notice that, by Barendregt's convention, none of the substitutions are defined on N.

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

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 '$\rightarrow_{\mathrm{XH}}$'.

Example 5.4 *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 Figure 2. We will see in Figure 3 how this reduction is modelled in the π-calculus through our interpretation.

ii) Reduction in '$\rightarrow_{\mathrm{XH}}$' is not deterministic in general:

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

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

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

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

(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,

$$zz\,\langle z := y\rangle\,\langle y := x\rangle\,\langle x := \Delta\rangle \ \rightarrow^*_{:=} \ yy\,\langle y := x\rangle\,\langle x := \Delta\rangle \ \rightarrow^*_{:=} \ xx\,\langle x := \Delta\rangle \ \rightarrow^*_{:=} \ \Delta\Delta$$

so, as stated by Lemma 5.3, the standard reduction result can be achieved by reduction in '$\rightarrow_{:=}$' (we will use Δ for $\lambda x.xx$ again below).

6 Interpreting $\lambda\mu\mathbf{x}$ in the π-calculus with pairing

We will now define our logical, output-based interpretation $\ulcorner M \lrcorner^{\mathrm{L}}\,a$ of the $\lambda\mu\mathbf{x}$-calculus into the π-calculus (where M is a $\lambda\mu\mathbf{x}$-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 Theorem 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 $\lambda\mu\mathbf{x}$ terms into the π-calculus is defined by:

Definition 6.1 (LOGICAL INTERPRETATION OF $\lambda\mu\mathbf{x}$ TERMS (*cf.* [13])) Let a not be a $\lambda\mu$-variable or name. Then

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

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

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

$$\begin{array}{rcll}
\ulcorner \mu\gamma.\mathit{Cmd} \lrcorner^{\mathsf{L}} a & \triangleq & (\nu\bullet)\,\ulcorner \mathit{Cmd} \lrcorner^{\mathsf{L}} \bullet\ \{a/\gamma\} & (\bullet\ \textit{fresh}) \\
\ulcorner [\beta]M \lrcorner^{\mathsf{L}} a & \triangleq & \ulcorner M \lrcorner^{\mathsf{L}} \beta &
\end{array}$$

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 Theorem 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 a single 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.[13]
- For the interpretation of an abstraction $(\nu xb)\,(\ulcorner M \lrcorner^{\mathsf{L}} b \mid \overline{a}\langle x,b\rangle)$, 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. It is possible to define $\ulcorner x := N \lrcorner^{\mathsf{L}} \triangleq \ulcorner N \lrcorner^{\mathsf{L}} x$, but this could cause not only the running of N *during* the substitution, but also synchronisations *between* substitution terms, which would not correspond to reductions in the domain, negatively affecting the completeness result.[14]

[13] This view of computation is exactly that of the calculus $\mathcal{X}$ when encoding the λ-calculus.

[14] The approach of [12] is to encode the variable x under a as $x(w).\overline{a}\,w$ and $\ulcorner x := N \lrcorner^{\mathsf{s}}$ through $!\ulcorner N \lrcorner^{\mathsf{s}} x$, which lets the encoding of N output directly on x.

- 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 $\ulcorner(\mu\alpha.[\beta]M)N\lrcorner^{\mathrm{L}} a \triangleq (\nu c)\,(\ulcorner M\lrcorner^{\mathrm{L}} \beta\,\{c/\alpha\} \mid !\,c(v,d).(\ulcorner v{:=}N\lrcorner^{\mathrm{L}} \mid !\,d{\rightarrow}\overline{a}))$ we need to be able to deal with the multiple outputs over c in $\ulcorner M\lrcorner^{\mathrm{L}} \beta\,\{c/\alpha\}$, 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.

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):

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

(see Theorem 4.8 in [11]).

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

$$\begin{array}{rcl} \ulcorner\mu\gamma.M\lrcorner^{\mathrm{L}} a & \triangleq & (\nu\bullet)\,\ulcorner M\lrcorner^{\mathrm{L}} \bullet\ \{a/\gamma\} \quad (\bullet\ \mathit{fresh}) \\ \ulcorner[\beta]M\lrcorner^{\mathrm{L}} a & \triangleq & \ulcorner M\lrcorner^{\mathrm{L}} \beta \end{array}$$

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

$$\ulcorner\mu\alpha.\lambda x.x\lrcorner^{\mathrm{L}} a \;=\; (\nu\bullet)\,((\nu xb)\,(x(u).!\,u{\rightarrow}\overline{b} \mid \overline{\bullet}\langle x,b\rangle))$$

is in normal form. Notice that all inputs and outputs are restricted; thereby, this process is weakly bisimilar to $\mathit{0}$ and to $\ulcorner\Delta\Delta\lrcorner^{\mathrm{L}} a$ (see Lemma 9.7). So using that interpretation, we cannot distinguish between *blocked* and *looping* computations. 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 $\alpha \neq \beta$):

$$\begin{array}{rclcl} \ulcorner\mu\alpha.[\beta]\lambda x.x\lrcorner^{\mathrm{L}} a & \triangleq & (\nu s)\,\ulcorner[\beta]\lambda x.x\lrcorner^{\mathrm{L}} s\,\{a/\alpha\} & \triangleq & (\nu s)\,\ulcorner\lambda x.x\lrcorner^{\mathrm{L}} \beta \qquad \equiv \\ & & \ulcorner\lambda x.x\lrcorner^{\mathrm{L}} \beta & \triangleq & (\nu xb)\,(\ulcorner x\lrcorner^{\mathrm{L}} b \mid \overline{\beta}\langle x,b\rangle) \end{array}$$

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

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

which is operationally the same as $\ulcorner M\lrcorner^{\mathrm{L}} \delta\,\{a/\gamma\}$ (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 (Theorem 7.9).

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

$$\begin{array}{rcl} \lceil \lambda yz.y \rfloor^{\mathrm{L}} a &=& (\nu yb)\,((\nu zd)\,(y(u).!\,u \rightarrow \overline{d} \mid \overline{b}\langle z,d\rangle) \mid \overline{a}\langle y,b\rangle) \\ \lceil \lambda x.x \rfloor^{\mathrm{L}} a &=& (\nu xb)\,(x(u).!\,u \rightarrow \overline{b} \mid \overline{a}\langle x,b\rangle) \end{array}$$

processes that differ under '$\approx$': 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 $(\nu x)\,(\lceil x := N \rfloor^{\mathrm{L}}) \approx \mathit{0}$, 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 $(\nu x)\,(!\,\lceil N \rfloor^{\mathrm{L}} x)$ – using the variant of [12] – *can have* observable behaviour, in contrast to here, where $(\nu x)\,(!\,\overline{x}(w).\lceil N \rfloor^{\mathrm{L}} w)$ *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 Theorem 7.5 and 12.1.

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

$$\lceil MN \rfloor^{\mathrm{s}} a \;\triangleq\; (\nu c)\,(\lceil M \rfloor^{\mathrm{H}} c \mid c(v,d).(\lceil \langle v := N \rangle \rfloor^{\mathrm{s}} \mid d \rightarrow \overline{a}))$$

where, in particular, the input on name c is *not replicated*: this corresponds to the fact that for λ-terms, in $\lceil M \rfloor^{\mathrm{H}} c$, 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 $\lceil \mu\alpha.[\alpha]M \rfloor^{\mathrm{L}} c = \lceil M \rfloor^{\mathrm{L}} \alpha\,\{c/\alpha\} = \lceil M\{c/\alpha\} \rfloor^{\mathrm{L}} c$, the output name c appears many times in the latter.

Remark 6.4 Observe the similarity between

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

The first communicates N via the main output channel c of M (which might occur more than once inside $\lceil M \rfloor^{\mathrm{L}} c$, 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.[15] In other words, application is just a special case of explicit structural substitution. As an abbreviation, we sometimes will write $(\nu c)\,(\lceil M \rfloor^{\mathrm{L}} c \mid \lceil c := N{\cdot}a \rfloor^{\mathrm{L}})$ for $\lceil MN \rfloor^{\mathrm{L}} a$.

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

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

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 $\lceil \langle \beta := N{\cdot}\gamma \rangle \rfloor^{\mathrm{L}}$, and we can use that $!\,P \approx\, !\,P \mid !\,P$.

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

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

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

Figure 3. Running $\lceil\!\lceil(\mu\alpha.[\alpha]\lambda y.y(\mu\delta.[\alpha]\lambda x.x)(\mu\delta'.[\alpha]\lambda x.x))(\lambda z.z)\rfloor\!\rfloor^{\mathrm{L}} a$ in '$\rightarrow_{\pi}$'.

$$\lceil\!\lceil \alpha := \lambda z.z\cdot a\rfloor\!\rfloor^{\mathrm{L}} \ \stackrel{\Delta}{=}\ !\,\alpha(v,d).(\lceil\!\lceil v := \lambda z.z\rfloor\!\rfloor^{\mathrm{L}} \mid !\,d\rightarrow\overline{a})$$

gets created directly by definition of $\lceil\!\lceil\cdot\rfloor\!\rfloor^{\mathrm{L}}\cdot$. This is used twice, in the steps marked $(*)$; the first exchanges the pair $\langle y,b\rangle$ over α which creates the process $\lceil\!\lceil y := \lambda z.z\rfloor\!\rfloor^{\mathrm{L}} \stackrel{\Delta}{=} !\,\overline{y}(w).\lceil\!\lceil\lambda z.z\rfloor\!\rfloor^{\mathrm{L}} w$ 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 $\lceil\!\lceil \alpha := \lambda z.z\cdot a\rfloor\!\rfloor^{\mathrm{L}}$ acts for the outermost application. The second use is again for a synchronisation over α, but now uses $\lceil\!\lceil \alpha := \lambda z.z\cdot a\rfloor\!\rfloor^{\mathrm{L}}$ 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 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 Theorem 13.8. In fact, the distributive character of structural substitution is dealt with entirely by congruence (see also Example 6.8).

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 puts into evidence 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 $\lceil \Delta\Delta \rfloor^{\mathsf{L}} a$ (perhaps the prototype of a non-typeable term) runs forever without generating output (see Example 9.1; this already holds for the interpretation of [12]).

Remark 6.5 Substitutions are interpreted as processes parallel to the main term:

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

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

We have, for example:

$$\begin{array}{ll} \lceil ((NM_1)S_1\cdots M_n)S_n \rfloor^{\mathsf{L}} a \;\triangleq\; (\nu \vec{y_n}\,\vec{\alpha_n})\,(\lceil (NM_1)S_1\cdots M_n \rfloor^{\mathsf{L}} a \mid \lceil S_n \rfloor^{\mathsf{L}}) & \triangleq \\ (\nu \vec{y_n}\,\vec{\alpha_n})\,((\nu c_1)\,(\lceil ((NM_1)S_1\cdots M_{n-1})S_{n-1} \rfloor^{\mathsf{L}} c_1 \mid \lceil c_1 := M_n\cdot a \rfloor^{\mathsf{L}}) \mid \lceil S_n \rfloor^{\mathsf{L}}) & \triangleq, \equiv \\ (\nu \vec{y_n}\,\vec{\alpha_n})\cdots(\nu \vec{y_1}\,\vec{\alpha_1})\,(\nu \vec{c}\,)\,(\lceil N \rfloor^{\mathsf{L}} c_1 \mid \lceil c_1 := M_1\cdot c_2 \rfloor^{\mathsf{L}} \mid \lceil S_1 \rfloor^{\mathsf{L}} \mid \cdots \mid \lceil c_n := M_n\cdot a \rfloor^{\mathsf{L}} \mid \lceil S_n \rfloor^{\mathsf{L}}) & \equiv \\ (\nu \vec{y_n}\,\vec{\alpha_n})\cdots(\nu \vec{y_1}\,\vec{\alpha_1})\,(\nu \vec{c}\,)\,(\lceil N \rfloor^{\mathsf{L}} c_1 \mid \lceil c_1 := M_1\cdot c_2 \rfloor^{\mathsf{L}} \mid \cdots \mid \lceil c_n := M_n\cdot a \rfloor^{\mathsf{L}} \mid \lceil S_1 \rfloor^{\mathsf{L}} \mid \cdots \mid \lceil S_n \rfloor^{\mathsf{L}}) & \equiv, \triangleq \\ \lceil (NM_1\cdots M_n)S_1\cdots S_n \rfloor^{\mathsf{L}} a & \end{array}$$

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.

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

i) If a is at most only used for output in $\lceil M \rfloor^{\mathsf{L}} g$ *and* $a \neq g$, *then* $(\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \lceil M \rfloor^{\mathsf{L}} g) \;\approx\; \lceil M \rfloor^{\mathsf{L}} g\,\{e/a\}$.

ii) If $a \notin M$, *then* $(\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \lceil M \rfloor^{\mathsf{L}} a) \;\approx\; \lceil M \rfloor^{\mathsf{L}} e$.

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

i) $M = x$: $(\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \lceil x \rfloor^{\mathsf{L}} g) \;\triangleq\; (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid x(u).!\,u{\rightarrow}\overline{g}) \;\equiv\; (\nu a)\,(!\,a{\rightarrow}\overline{e}) \mid x(u).!\,u{\rightarrow}\overline{g} \;\approx$
$x(u).!\,u{\rightarrow}\overline{g} \;\triangleq\; \lceil x \rfloor^{\mathsf{L}} g \;=\; \lceil x \rfloor^{\mathsf{L}} g\,\{e/a\}$

$M = \lambda x.N$: $(\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \lceil \lambda x.N \rfloor^{\mathsf{L}} g) \;\triangleq\; (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid (\nu xb)\,(\lceil N \rfloor^{\mathsf{L}} b \mid \overline{g}\langle x,b\rangle)) \;\equiv\; (a \neq g)$
$(\nu xb)\,((\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \lceil N \rfloor^{\mathsf{L}} b) \mid \overline{g}\langle x,b\rangle) \;\approx (IH)\; (\nu xb)\,(\lceil N \rfloor^{\mathsf{L}} b\,\{e/a\} \mid \overline{g}\langle x,b\rangle) \;\triangleq$
$(\nu xb)\,(\lceil N \rfloor^{\mathsf{L}} b \mid \overline{g}\langle x,b\rangle)\,\{e/a\} \;\triangleq (a \neq g)\; \lceil \lambda x.N \rfloor^{\mathsf{L}} g\,\{e/a\}$

$$\begin{array}{lll} M = PQ\colon\; (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \lceil PQ \rfloor^{\mathsf{L}} g) & \triangleq & \\ (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid (\nu c)\,(\lceil P \rfloor^{\mathsf{L}} c \mid !\,c(v,d).(!\,\overline{v}(w).\lceil Q \rfloor^{\mathsf{L}} w \mid !\,d{\rightarrow}\overline{g}))) & \approx & (2.5{:}1) \\ (\nu c)\,((\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \lceil P \rfloor^{\mathsf{L}} c) \mid (\nu a)\,(!\,a{\rightarrow}\overline{e} \mid !\,c(v,d).(!\,\overline{v}(w).\lceil Q \rfloor^{\mathsf{L}} w \mid !\,d{\rightarrow}\overline{g}))) & \approx & (2.5{:}4) \\ (\nu c)\,((\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \lceil P \rfloor^{\mathsf{L}} c) \mid !\,c(v,d).((\nu a)\,(!\,a{\rightarrow}\overline{e} \mid !\,\overline{v}(w).\lceil Q \rfloor^{\mathsf{L}} w \mid !\,d{\rightarrow}\overline{g}))) & \approx & (2.5{:}5) \\ (\nu c)\,((\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \lceil P \rfloor^{\mathsf{L}} c) \mid !\,c(v,d).(!\,\overline{v}(w).(\nu a)\,(!\,a{\rightarrow}\overline{e} \mid \lceil Q \rfloor^{\mathsf{L}} w) \mid !\,d{\rightarrow}\overline{g})) & \approx & (IH) \\ (\nu c)\,(\lceil P \rfloor^{\mathsf{L}} c\,\{e/a\} \mid !\,c(v,d).(!\,\overline{v}(w).\lceil Q \rfloor^{\mathsf{L}} w\,\{e/a\} \mid !\,d{\rightarrow}\overline{g})) & \triangleq & (a \neq c,g) \\ (\nu c)\,(\lceil P \rfloor^{\mathsf{L}} c \mid !\,c(v,d).(\lceil v := Q \rfloor^{\mathsf{L}} \mid !\,d{\rightarrow}\overline{g}))\,\{e/a\} & \triangleq & \lceil PQ \rfloor^{\mathsf{L}} g\,\{e/a\} \end{array}$$

$$\begin{array}{lll}
M = P\langle x := Q\rangle:\ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner P\langle x := Q\rangle \lrcorner^{\mathsf{L}} g) & \triangleq & \\
\quad (\nu a)\,(!\,a\rightarrow\overline{e} \mid (\nu x)\,(\ulcorner P \lrcorner^{\mathsf{L}} g \mid !\,\overline{x}(w).\ulcorner Q \lrcorner^{\mathsf{L}} w)) & \approx & (2.5{:}1,5) \\
\quad (\nu x)\,((\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner P \lrcorner^{\mathsf{L}} g) \mid !\,\overline{x}(w).(\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner Q \lrcorner^{\mathsf{L}} w)) & \approx & (IH) \\
\quad (\nu x)\,(\ulcorner P \lrcorner^{\mathsf{L}} g\,\{e/a\} \mid !\,\overline{x}(w).\ulcorner Q \lrcorner^{\mathsf{L}} w\,\{e/a\}) & \triangleq & \\
\quad (\nu x)\,(\ulcorner P \lrcorner^{\mathsf{L}} g \mid !\,\overline{x}(w).\ulcorner Q \lrcorner^{\mathsf{L}} w)\,\{e/a\} & \triangleq & \ulcorner P\langle x := Q\rangle \lrcorner^{\mathsf{L}} g\,\{e/a\}
\end{array}$$

$$M = \mu\beta.[\beta]N:\ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner \mu\beta.[\beta]N \lrcorner^{\mathsf{L}} g) \ \triangleq\ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner N \lrcorner^{\mathsf{L}} \beta\,\{g/\beta\}) \ \approx\ (IH)$$
$$\quad \ulcorner N \lrcorner^{\mathsf{L}} \beta\,\{e/a\}\,\{g/\beta\} \ \triangleq\ \ulcorner \mu\beta.[\beta]N \lrcorner^{\mathsf{L}} g\,\{e/a\}$$

$$M = \mu\beta.[\gamma]N,\ \beta \neq \gamma:\ \ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner \mu\beta.[\gamma]N \lrcorner^{\mathsf{L}} g) \ \triangleq\ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner N \lrcorner^{\mathsf{L}} \gamma\,\{g/\beta\}) \ \approx\ (IH)$$
$$\quad \ulcorner N \lrcorner^{\mathsf{L}} \gamma\,\{e/a\}\,\{g/\beta\} \ = (a \neq g)\ \ \ulcorner N \lrcorner^{\mathsf{L}} \gamma\,\{g/\beta\}\,\{e/a\} \ \triangleq\ \ulcorner \mu\beta.[\gamma]N \lrcorner^{\mathsf{L}} g\,\{e/a\}$$

$$\begin{array}{lll}
M = P\langle \beta := Q{\cdot}\gamma\rangle:\ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner P\langle \beta := Q{\cdot}\gamma\rangle \lrcorner^{\mathsf{L}} g) & \triangleq & \\
\quad (\nu a)\,(!\,a\rightarrow\overline{e} \mid (\nu\beta)\,(\ulcorner P \lrcorner^{\mathsf{L}} g \mid !\,\beta(v,d).(!\,\overline{v}(w).\ulcorner Q \lrcorner^{\mathsf{L}} w \mid !\,d\rightarrow\gamma))) & \approx & (2.5{:}1,4) \\
\quad (\nu\beta)\,((\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner P \lrcorner^{\mathsf{L}} g) \mid !\,\beta(v,d).(!\,\overline{v}(w).(\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner Q \lrcorner^{\mathsf{L}} w) \mid !\,d\rightarrow\gamma)) & \approx & (IH) \\
\quad (\nu\beta)\,(\ulcorner P \lrcorner^{\mathsf{L}} g\,\{e/a\} \mid !\,\beta(v,d).(!\,\overline{v}(w).\ulcorner Q \lrcorner^{\mathsf{L}} w\,\{e/a\} \mid !\,d\rightarrow\gamma)) & \triangleq & \\
\quad (\nu\beta)\,(\ulcorner P \lrcorner^{\mathsf{L}} g \mid !\,\beta(v,d).(\ulcorner v := Q \lrcorner^{\mathsf{L}} \mid !\,d\rightarrow\overline{\gamma}))\,\{e/a\} & \triangleq & \\
\quad \ulcorner P\langle \beta := Q{\cdot}\gamma\rangle \lrcorner^{\mathsf{L}} g\,\{e/a\} & &
\end{array}$$

ii) $M = x$: $(\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner x \lrcorner^{\mathsf{L}} a) \ \triangleq\ (\nu a)\,(!\,a\rightarrow\overline{e} \mid x(u).!\,u\rightarrow\overline{a}) \ \approx (2.5{:}3)$

$$\quad x(u).((\nu a)\,(!\,a\rightarrow\overline{e} \mid !\,u\rightarrow\overline{a})) \ \approx x(u).!\,u\rightarrow\overline{e} \ \triangleq\ \ulcorner x \lrcorner^{\mathsf{L}} e$$

$$\begin{array}{lll}
M = \lambda x.N:\ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner \lambda x.N \lrcorner^{\mathsf{L}} a) & \triangleq & (\nu a)\,(!\,a\rightarrow\overline{e} \mid (\nu xb)\,(\ulcorner N \lrcorner^{\mathsf{L}} b \mid \overline{a}\langle x,b\rangle)) \ \equiv \\
\quad (\nu axb)\,(!\,a\rightarrow\overline{e} \mid \ulcorner N \lrcorner^{\mathsf{L}} b \mid \overline{a}\langle x,b\rangle) & \equiv & (a \notin \ulcorner N \lrcorner^{\mathsf{L}} b) \\
\quad (\nu xb)\,(\ulcorner N \lrcorner^{\mathsf{L}} b \mid (\nu a)\,(!\,a\rightarrow\overline{e} \mid \overline{a}\langle x,b\rangle)) & \approx & (2.6{:}11)\,(\nu xb)\,(\ulcorner N \lrcorner^{\mathsf{L}} b \mid \overline{e}\langle x,b\rangle) \quad \triangleq\ \ulcorner \lambda x.N \lrcorner^{\mathsf{L}} e
\end{array}$$

$$\begin{array}{lll}
M = PQ:\ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner PQ \lrcorner^{\mathsf{L}} a) & \triangleq & \\
\quad (\nu a)\,(!\,a\rightarrow\overline{e} \mid (\nu c)\,(\ulcorner P \lrcorner^{\mathsf{L}} c \mid !\,c(v,d).(!\,\overline{v}(w).\ulcorner Q \lrcorner^{\mathsf{L}} w \mid !\,d\rightarrow\overline{a}))) & \approx & (2.5) \\
\quad (\nu c)\,((\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner P \lrcorner^{\mathsf{L}} c) \mid !\,c(v,d).(!\,\overline{v}(w).(\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner Q \lrcorner^{\mathsf{L}} w) \mid (\nu a)\,(!\,a\rightarrow\overline{e} \mid !\,d\rightarrow\overline{a}))) & \approx & \\
 & & (a \notin \ulcorner P \lrcorner^{\mathsf{L}} c, \ulcorner Q \lrcorner^{\mathsf{L}} w) \\
\quad (\nu c)\,((\nu a)\,(!\,a\rightarrow\overline{e}) \mid \ulcorner P \lrcorner^{\mathsf{L}} c \mid !\,c(v,d).(!\,\overline{v}(w).(\nu a)\,(!\,a\rightarrow\overline{e}) \mid \ulcorner Q \lrcorner^{\mathsf{L}} w \mid (\nu a)\,(!\,a\rightarrow\overline{e} \mid !\,d\rightarrow\overline{a}))) & \approx & \\
\quad (\nu c)\,(\ulcorner P \lrcorner^{\mathsf{L}} c \mid !\,c(v,d).(!\,\overline{v}(w).\ulcorner Q \lrcorner^{\mathsf{L}} w \mid !\,d\rightarrow\overline{e})) & \triangleq & \ulcorner PQ \lrcorner^{\mathsf{L}} e
\end{array}$$

$$\begin{array}{lll}
M = P\langle x := Q\rangle:\ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner P\langle x := Q\rangle \lrcorner^{\mathsf{L}} a) & \triangleq & \\
\quad (\nu a)\,(!\,a\rightarrow\overline{e} \mid (\nu x)\,(\ulcorner P \lrcorner^{\mathsf{L}} a \mid !\,\overline{x}(w).\ulcorner Q \lrcorner^{\mathsf{L}} w)) & \equiv & (x \neq a,\ a \notin !\,\overline{x}(w).\ulcorner Q \lrcorner^{\mathsf{L}} w) \\
\quad (\nu x)\,((\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner P \lrcorner^{\mathsf{L}} a) \mid !\,\overline{x}(w).\ulcorner Q \lrcorner^{\mathsf{L}} w) & \approx & (IH) \\
\quad (\nu x)\,(\ulcorner P \lrcorner^{\mathsf{L}} e \mid !\,\overline{x}(w).\ulcorner Q \lrcorner^{\mathsf{L}} w) & \triangleq & \ulcorner P\langle x := Q\rangle \lrcorner^{\mathsf{L}} e
\end{array}$$

$$M = \mu\beta.[\beta]N:\ \ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner \mu\beta.[\beta]N \lrcorner^{\mathsf{L}} a) \ \triangleq\ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner N \lrcorner^{\mathsf{L}} \beta\,\{a/\beta\}) \ =$$
$$\quad (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner N \lrcorner^{\mathsf{L}} a) \ \approx (IH)\ \ \ulcorner N \lrcorner^{\mathsf{L}} e \ = \ \ulcorner N \lrcorner^{\mathsf{L}} \beta\,\{e/\beta\} \ \triangleq\ \ulcorner \mu\beta.[\beta]N \lrcorner^{\mathsf{L}} e$$

$$M = \mu\beta.[\gamma]N,\ \beta \neq \gamma:\ \ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner \mu\beta.[\gamma]N \lrcorner^{\mathsf{L}} a) \ \triangleq\ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner N \lrcorner^{\mathsf{L}} \gamma\,\{a/\beta\}) \ =_{\alpha}$$
$$\quad (\nu\beta)\,(!\,\beta\rightarrow\overline{e} \mid \ulcorner N \lrcorner^{\mathsf{L}} \gamma) \ \approx (\mathit{part}\ (i))\ \ \ulcorner N \lrcorner^{\mathsf{L}} \gamma\,\{e/\beta\} \ \triangleq\ \ulcorner \mu\beta.[\gamma]N \lrcorner^{\mathsf{L}} e$$

$$\begin{array}{lll}
M = P\langle \beta := Q{\cdot}\gamma\rangle:\ (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner P\langle \beta := Q{\cdot}\gamma\rangle \lrcorner^{\mathsf{L}} a) & \triangleq & \\
\quad (\nu a)\,(!\,a\rightarrow\overline{e} \mid (\nu\beta)\,(\ulcorner P \lrcorner^{\mathsf{L}} a \mid \ulcorner \beta := Q{\cdot}\gamma \lrcorner^{\mathsf{L}})) & \approx & (2.5{:}1) \\
\quad (\nu\beta)\,((\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner P \lrcorner^{\mathsf{L}} a) \mid (\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner \beta := Q{\cdot}\gamma \lrcorner^{\mathsf{L}})) & \equiv & (a \notin \ulcorner \beta := Q{\cdot}\gamma \lrcorner^{\mathsf{L}}) \\
\quad (\nu\beta)\,((\nu a)\,(!\,a\rightarrow\overline{e} \mid \ulcorner P \lrcorner^{\mathsf{L}} a) \mid (\nu a)\,(!\,a\rightarrow\overline{e}) \mid \ulcorner \beta := Q{\cdot}\gamma \lrcorner^{\mathsf{L}}) & \approx & (IH) \\
\quad (\nu\beta)\,(\ulcorner P \lrcorner^{\mathsf{L}} e \mid \ulcorner \beta := Q{\cdot}\gamma \lrcorner^{\mathsf{L}}) & \triangleq & \ulcorner P\langle \beta := Q{\cdot}\gamma\rangle \lrcorner^{\mathsf{L}} e \qquad \Box
\end{array}$$

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

Definition 6.7 *i)* We define a *garbage collection* bisimilarity by: $P \approx_{\mathrm{G}} Q$ if and only if there exists R such that $P \equiv Q \,|\, R$ and $R \approx 0$. We call a process that is weakly bisimilar to 0 *garbage*.

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

a) for all M: $(\nu a)\,(\lceil M \rfloor^{\mathrm{L}} a \,|\, !\, a \rightarrow \overline{e}) \approx_{\mathrm{R}} \lceil M \rfloor^{\mathrm{L}} e$.

b) for all M: if $a \neq b$, then $(\nu a)\,(\lceil M \rfloor^{\mathrm{L}} b \,|\, !\, a \rightarrow \overline{e}) \approx_{\mathrm{R}} \lceil M \rfloor^{\mathrm{L}} b\,\{e/a\}$.

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

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

a) for all M, N: $(\nu \alpha)\,(\lceil M \rfloor^{\mathrm{L}} \alpha \,|\, !\, \alpha(v,d).(\lceil v := N \rfloor^{\mathrm{L}} \,|\, !\, d \rightarrow \overline{\gamma})) \;\approx_{\mathrm{D}}$

$$(\nu \alpha)\,((\nu c)\,(\lceil M \rfloor^{\mathrm{L}} c \,|\, !\, c(v,d).(\lceil v := N \rfloor^{\mathrm{L}} \,|\, !\, d \rightarrow \overline{\gamma})) \,|\, !\, \alpha(v,d).(\lceil v := N \rfloor^{\mathrm{L}} \,|\, !\, d \rightarrow \overline{\gamma}))\ \ (c\ \mathit{fresh})$$

Notice that we 'split' the substitution $!\, \alpha(v,d).(\lceil v := N \rfloor^{\mathrm{L}} \,|\, !\, d \rightarrow \overline{\gamma})$ 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 $\lceil M \rfloor^{\mathrm{L}} c$.

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

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 6.6, and that '$\approx_{\mathrm{D}}$' $\subset$ '$\approx$' by Theorem 2.5:2.

Using the Renaming Lemma 6.6, we can show the following:

Example 6.8 The interpretation of the β-redex $(\lambda x.M)N$ reduces as follows:

$$\begin{array}{lll}
\lceil (\lambda x.M)N \rfloor^{\mathrm{L}} a & \triangleq & \\
(\nu c)\,((\nu xb)\,(\lceil M \rfloor^{\mathrm{L}} b \,|\, \overline{c}\langle x,b\rangle) \,|\, !\, c(v,d).(\lceil v := N \rfloor^{\mathrm{L}} \,|\, !\, d \rightarrow \overline{a})) & \rightarrow_{\pi} (c) & (c \notin fn(M,N)) \\
(\nu bx)\,(\lceil M \rfloor^{\mathrm{L}} b \,|\, !\, b \rightarrow \overline{a} \,|\, \lceil x := N \rfloor^{\mathrm{L}}) \,|\, (\nu c)\,(!\, c(v,d).(\lceil v := N \rfloor^{\mathrm{L}} \,|\, !\, d \rightarrow \overline{a})) & \approx_{\mathrm{R}} & (6.6) \\
(\nu x)\,(\lceil M \rfloor^{\mathrm{L}} a \,|\, \lceil x := N \rfloor^{\mathrm{L}}) \,|\, (\nu c)\,(!\, c(v,d).(\lceil v := N \rfloor^{\mathrm{L}} \,|\, !\, d \rightarrow \overline{a})) & \approx_{\mathrm{G}} & (*) \\
(\nu x)\,(\lceil M \rfloor^{\mathrm{L}} a \,|\, \lceil x := N \rfloor^{\mathrm{L}}) \;\triangleq\; \lceil M\,\langle x := N\rangle \rfloor^{\mathrm{L}} a & &
\end{array}$$

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

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

Lemma 6.9 (SUBSTITUTION LEMMA) $\lceil M\,\langle y := N\rangle\,\langle x := L\rangle \rfloor^{\mathrm{L}} a \;\approx\; \lceil M\,\langle x := L\rangle\,\langle y := N\,\langle x := L\rangle\rangle \rfloor^{\mathrm{L}} a$.

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

$$\begin{array}{lll}
\lceil M\,\langle y := N\rangle\,\langle x := L\rangle \rfloor^{\mathrm{L}} a & \triangleq & \\
(\nu x)\,((\nu y)\,(\lceil M \rfloor^{\mathrm{L}} a \,|\, !\, \overline{y}(w).\lceil N \rfloor^{\mathrm{L}} w) \,|\, !\, \overline{x}(w).\lceil L \rfloor^{\mathrm{L}} w) & \approx & (2.5{:}6) \\
(\nu y)\,((\nu x)\,(\lceil M \rfloor^{\mathrm{L}} a \,|\, !\, \overline{x}(w).\lceil L \rfloor^{\mathrm{L}} w) \,|\, (\nu x)\,(!\, \overline{y}(w).\lceil N \rfloor^{\mathrm{L}} w \,|\, !\, \overline{x}(w).\lceil L \rfloor^{\mathrm{L}} w)) & \approx & (2.5{:}9) \\
(\nu y)\,((\nu x)\,(\lceil M \rfloor^{\mathrm{L}} a \,|\, !\, \overline{x}(w).\lceil L \rfloor^{\mathrm{L}} w) \,|\, !\, \overline{y}(w).(\nu x)\,(\lceil N \rfloor^{\mathrm{L}} y \,|\, !\, \overline{x}(w).\lceil L \rfloor^{\mathrm{L}} w)) & \triangleq & \\
(\nu y)\,(\lceil M\,\langle x := L\rangle \rfloor^{\mathrm{L}} a \,|\, !\, \overline{y}(w).\lceil N\,\langle x := L\rangle \rfloor^{\mathrm{L}} w) & \triangleq & \\
\lceil M\,\langle x := L\rangle\,\langle y := N\,\langle x := L\rangle\rangle \rfloor^{\mathrm{L}} a & &
\end{array}$$

□

7 Soundness, completeness, and termination

As in [43, 51], 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 $\lceil \cdot \rfloor^{\mathrm{L}} \cdot$ preserves '$\rightarrow_{\mathrm{XH}}$' 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 $P \rightarrow_{\mathrm{XH}} Q$, then there exist R such that $\lceil P \rfloor^{\mathrm{L}} a \rightarrow^*_\pi R$ and $R \approx_{\mathrm{RGD}} \lceil Q \rfloor^{\mathrm{L}} a$ (*i.e.* $\lceil P \rfloor^{\mathrm{L}} a \rightarrow^*_\pi, \approx_{\mathrm{RGD}} \lceil Q \rfloor^{\mathrm{L}} a$).*

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

Main reduction rules : (β): Then $P \equiv (\lambda x.M)\,N$ and $Q \equiv M\,\langle x := N \rangle$; by Example 6.8.

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

$$\begin{array}{ll}
\lceil (\mu\alpha.[\alpha]\,M)\,N \rfloor^{\mathrm{L}} a \;\stackrel{\Delta}{=}\; (\nu c)\,(\lceil \mu\alpha.[\alpha]\,M \rfloor^{\mathrm{L}} c \mid \lceil c := N\cdot a \rfloor^{\mathrm{L}}) & \stackrel{\Delta}{=} \\
(\nu c)\,(\lceil M \rfloor^{\mathrm{L}} \alpha\,\{c/\alpha\} \mid !\,c(v,d).(\lceil v := N \rfloor^{\mathrm{L}} \mid !\,d \rightarrow \overline{a})) & \approx_{\mathrm{D}} \\
(\nu c)\,((\nu\alpha)\,(\lceil M \rfloor^{\mathrm{L}} c \mid \lceil \alpha := N\cdot a \rfloor^{\mathrm{L}}) \mid \lceil c := N\cdot a \rfloor^{\mathrm{L}}) & = \\
(\nu c)\,((\nu\alpha)\,(\lceil M \rfloor^{\mathrm{L}} c \mid \lceil \alpha := N\cdot\gamma \rfloor^{\mathrm{L}}) \mid \lceil c := N\cdot\gamma \rfloor^{\mathrm{L}})\,\{a/\gamma\} & \stackrel{\Delta}{=} \\
(\nu c)\,(\lceil M\,\langle \alpha := N\cdot\gamma \rangle \rfloor^{\mathrm{L}} c \mid \lceil c := N\cdot\gamma \rfloor^{\mathrm{L}})\,\{a/\gamma\} & \stackrel{\Delta}{=} \\
\lceil M\,\langle \alpha := N\cdot\gamma \rangle\,N \rfloor^{\mathrm{L}} \gamma\,\{a/\gamma\} \;\stackrel{\Delta}{=}\; \lceil \mu\gamma.[\gamma]\,M\,\langle \alpha := N\cdot\gamma \rangle\,N \rfloor^{\mathrm{L}} a &
\end{array}$$

(μ_r): Then $P \equiv (\mu\alpha.[\beta]\,M)\,N$ and $Q \equiv \mu\alpha.[\beta]\,M\,\langle \alpha := N\cdot\gamma \rangle\,N$, with $\alpha \neq \beta$, γ fresh, and

$$\begin{array}{lcll}
\lceil (\mu\alpha.[\beta]\,M)\,N \rfloor^{\mathrm{L}} a & \stackrel{\Delta}{=} & (\nu c)\,(\lceil \mu\alpha.[\beta]\,M \rfloor^{\mathrm{L}} c \mid \lceil c := N\cdot a \rfloor^{\mathrm{L}}) & \stackrel{\Delta}{=} \\
(\nu c)\,(\lceil M \rfloor^{\mathrm{L}} \beta\,\{c/\alpha\} \mid \lceil c := N\cdot a \rfloor^{\mathrm{L}}) & =_\alpha & (c\;\mathit{fresh})(\nu\alpha)\,(\lceil M \rfloor^{\mathrm{L}} \beta \mid \lceil \alpha := N\cdot\gamma \rfloor^{\mathrm{L}})\,\{a/\gamma\} & \stackrel{\Delta}{=} \\
\lceil M\,\langle \alpha := N\cdot\gamma \rangle \rfloor^{\mathrm{L}} \beta\,\{a/\gamma\} & \stackrel{\Delta}{=} & \lceil \mu\gamma.[\beta]\,M\,\langle \alpha := N\cdot\gamma \rangle \rfloor^{\mathrm{L}} a &
\end{array}$$

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

$$\begin{array}{ll}
\delta = \gamma: & \lceil \mu\alpha.[\beta]\mu\gamma.[\gamma]\,M \rfloor^{\mathrm{L}} a \;\stackrel{\Delta}{=}\; \lceil \mu\gamma.[\gamma]M \rfloor^{\mathrm{L}} \beta\,\{a/\alpha\} \;\stackrel{\Delta}{=}\; \lceil M \rfloor^{\mathrm{L}} \gamma\,\{\beta/\gamma\}\,\{a/\alpha\} \;= \\
& \lceil M\,\{\beta/\gamma\} \rfloor^{\mathrm{L}} \beta\,\{a/\alpha\} \;\stackrel{\Delta}{=}\; \lceil \mu\alpha.[\beta]\,M\,\{\beta/\gamma\} \rfloor^{\mathrm{L}} a \\[1ex]
\delta \neq \gamma: & \lceil \mu\alpha.[\beta]\mu\gamma.[\delta]\,M \rfloor^{\mathrm{L}} a \;\stackrel{\Delta}{=}\; \lceil \mu\gamma.[\delta]M \rfloor^{\mathrm{L}} \beta\,\{a/\alpha\} \;\stackrel{\Delta}{=}\; \lceil M \rfloor^{\mathrm{L}} \delta\,\{\beta/\gamma\}\,\{a/\alpha\} \;= \\
& \lceil M\,\{\beta/\gamma\} \rfloor^{\mathrm{L}} \delta\,\{a/\alpha\} \;\stackrel{\Delta}{=}\; \lceil \mu\alpha.[\delta]\,M\,\{\beta/\gamma\} \rfloor^{\mathrm{L}} a
\end{array}$$

(C): Then $P \equiv \mu\alpha.[\alpha]\,M$ and $Q \equiv M$, with $\alpha \notin \mathit{fn}(M)$, and

$$\lceil \mu\alpha.[\alpha]\,M \rfloor^{\mathrm{L}} a \;\stackrel{\Delta}{=}\; \lceil M \rfloor^{\mathrm{L}} \alpha\,\{a/\alpha\} \;=(\alpha \notin \mathit{fn}(M))\; \lceil M \rfloor^{\mathrm{L}} a\;.$$

Substitution rules : (hv): Then $P \equiv xS_0M_1S_1\cdots M_nS_n$ and $Q \equiv NS_0M_1S_1\cdots M_nS_n$ provided $\langle x := N \rangle \in S_n$. By Remark 6.5, we can consider the substitution be placed on the outside; take $S = S_0\cdots S_n$.

$$\begin{array}{ll}
\lceil xS_0M_1S_1\cdots M_nS_n \rfloor^{\mathrm{L}} a \quad\equiv (6.5)\quad \lceil xM_1\cdots M_nS \rfloor^{\mathrm{L}} a & \stackrel{\Delta}{=} \\
(\nu \overrightarrow{y_n}\,\overrightarrow{\alpha_n}\,)\,(\lceil xM_1\cdots M_n \rfloor^{\mathrm{L}} a \mid \lceil \boldsymbol{S} \rfloor^{\mathrm{L}}) & \stackrel{\Delta}{=} \\
(\nu \overrightarrow{y_n}\,\overrightarrow{\alpha_n}\,)\,((\nu c)\,(\lceil xM_1\cdots M_{n-1} \rfloor^{\mathrm{L}} c \mid \lceil c := M_n\cdot a \rfloor^{\mathrm{L}}) \mid \lceil \boldsymbol{S} \rfloor^{\mathrm{L}}) & \stackrel{\Delta}{=} \quad (c_{n+1} = a) \\
(\nu \overrightarrow{y_n}\,\overrightarrow{\alpha_n}\,)\,(\lceil x \rfloor^{\mathrm{L}} c_1 \mid \overrightarrow{\lceil c_i := M_i\cdot c_{i+1} \rfloor^{\mathrm{L}}} \mid \lceil \boldsymbol{S} \rfloor^{\mathrm{L}}) & \stackrel{\Delta}{=}, \equiv \quad (!\,\overline{x}(w).\lceil N \rfloor^{\mathrm{L}} w \in \lceil \boldsymbol{S} \rfloor^{\mathrm{L}}) \\
(\nu \overrightarrow{y_n}\,\overrightarrow{\alpha_n}\,)\,(x(u).!\,u \rightarrow \overline{c_1} \mid \overrightarrow{\lceil c_i := M_i\cdot c_{i+1} \rfloor^{\mathrm{L}}} \mid \overline{x}(w).\lceil N \rfloor^{\mathrm{L}} w \mid \lceil \boldsymbol{S} \rfloor^{\mathrm{L}}) & \rightarrow_\pi (x) \\
(\nu \overrightarrow{y_n}\,\overrightarrow{\alpha_n}\,)\,((\nu w)\,(\lceil N \rfloor^{\mathrm{L}} w \mid !\,w \rightarrow \overline{c_1}) \mid \overrightarrow{\lceil c_i := M_i\cdot c_{i+1} \rfloor^{\mathrm{L}}} \mid \lceil \boldsymbol{S} \rfloor^{\mathrm{L}}) & \approx_{\mathrm{R}} \\
(\nu \overrightarrow{y_n}\,\overrightarrow{\alpha_n}\,)\,(\lceil N \rfloor^{\mathrm{L}} c_1 \mid \overrightarrow{\lceil c_i := M_i\cdot c_{i+1} \rfloor^{\mathrm{L}}} \mid \lceil \boldsymbol{S} \rfloor^{\mathrm{L}}) & \stackrel{\Delta}{=} \\
\lceil NM_1\cdots M_nS \rfloor^{\mathrm{L}} a \quad\equiv\quad \lceil NS_0M_1S_1\cdots M_nS_n \rfloor^{\mathrm{L}} a &
\end{array}$$

(λS): Then $P \equiv (\lambda y.M)S$ and $Q \equiv \lambda x.MS$, and $\quad \lceil (\lambda x.M)S \rfloor^{\mathrm{L}} a \;\stackrel{\Delta}{=}$

$$(\nu \overrightarrow{y\alpha}\,)\,((\nu xb)\,(\lceil M \rfloor^{\mathrm{L}} b \mid \overline{a}\langle x,b \rangle) \mid \lceil \boldsymbol{S} \rfloor^{\mathrm{L}}) \;\equiv (a \notin \overrightarrow{y\alpha}\,)\; (\nu xb)\,((\nu \overrightarrow{y\alpha}\,)\,(\lceil M \rfloor^{\mathrm{L}} b \mid \lceil \boldsymbol{S} \rfloor^{\mathrm{L}}) \mid \overline{a}\langle y,b \rangle) \;\stackrel{\Delta}{=}\; \lceil \lambda x.MS \rfloor^{\mathrm{L}} a$$

(hn): Then $P \equiv (\mu\delta.[\beta]M)S$ and $Q \equiv (\mu\delta.[\gamma]M\,\langle \beta := N\cdot\gamma \rangle\,N)\,S\backslash\beta$, with $\langle \beta := N\cdot\gamma \rangle \in S$, and

$$\begin{array}{lll}
\lceil(\mu\delta.[\beta]M)S\rfloor^{\mathrm{L}}a \quad \triangleq \quad (\nu\overrightarrow{y\alpha})\,(\lceil M\rfloor^{\mathrm{L}}\beta\,\{a/\delta\} \mid \lceil \boldsymbol{S}\rfloor^{\mathrm{L}}) & \equiv & (\lceil\beta:=N\cdot\gamma\rfloor^{\mathrm{L}} \in \lceil\boldsymbol{S}\rfloor^{\mathrm{L}}) \\
(\nu\overrightarrow{y\alpha})\,((\nu\beta)\,(\lceil M\rfloor^{\mathrm{L}}\beta\,\{a/\delta\} \mid !\,\beta(v,d).(\lceil v:=N\rfloor^{\mathrm{L}} \mid !\,d\rightarrow\overline{\gamma})) \mid \lceil S\backslash\beta\rfloor^{\mathrm{L}}) & = & (\delta \notin \lceil\beta:=N\cdot\gamma\rfloor^{\mathrm{L}}) \\
(\nu\overrightarrow{y\alpha})\,((\nu\beta)\,(\lceil M\rfloor^{\mathrm{L}}\beta \mid \lceil\beta:=N\cdot\gamma\rfloor^{\mathrm{L}})\,\{a/\delta\} \mid \lceil S\backslash\beta\rfloor^{\mathrm{L}}) & \approx_{\mathrm{D}} & \\
(\nu\overrightarrow{y\alpha})\,((\nu c)\,((\nu\beta)\,(\lceil M\rfloor^{\mathrm{L}}c \mid \lceil\beta:=N\cdot\gamma\rfloor^{\mathrm{L}}) \mid \lceil c:=N\cdot\gamma\rfloor^{\mathrm{L}})\,\{a/\delta\} \mid \lceil S\backslash\beta\rfloor^{\mathrm{L}}) & \triangleq & \\
(\nu\overrightarrow{y\alpha})\,((\nu c)\,(\lceil M\langle\beta:=N\cdot\gamma\rangle\rfloor^{\mathrm{L}}c \mid \lceil c:=N\cdot\gamma\rfloor^{\mathrm{L}})\,\{a/\delta\} \mid \lceil S\backslash\alpha\rfloor^{\mathrm{L}}) & \triangleq & \\
(\nu\overrightarrow{y\alpha})\,(\lceil(M\langle\beta:=N\cdot\gamma\rangle)N\rfloor^{\mathrm{L}}\gamma\,\{a/\delta\} \mid \lceil S\backslash\beta\rfloor^{\mathrm{L}}) & \triangleq & \\
(\nu\overrightarrow{y\alpha})\,(\lceil\mu\delta.[\gamma](M\langle\beta:=N\cdot\gamma\rangle)N\rfloor^{\mathrm{L}}a \mid \lceil S\backslash\beta\rfloor^{\mathrm{L}}) & \triangleq & \\
\lceil(\mu\delta.[\gamma](M\langle\beta:=N\cdot\gamma\rangle)N)\,S\backslash\beta\rfloor^{\mathrm{L}}a & &
\end{array}$$

(nS): Then $P \equiv (\mu\delta.[\beta]M)S$ and $Q \equiv \mu\delta.[\beta]MS$, provided $\beta \notin S$, and $\lceil(\mu\delta.[\beta]M)S\rfloor^{\mathrm{L}}a \triangleq (\nu\overrightarrow{y\alpha})\,(\lceil M\rfloor^{\mathrm{L}}\beta\,\{a/\delta\} \mid \lceil\boldsymbol{S}\rfloor^{\mathrm{L}}) = (\delta\notin S)\ (\nu\overrightarrow{y\alpha})\,(\lceil M\rfloor^{\mathrm{L}}\beta \mid \lceil\boldsymbol{S}\rfloor^{\mathrm{L}})\{a/\delta\} \triangleq \lceil MS\rfloor^{\mathrm{L}}\beta\,\{a/\delta\} \triangleq \lceil\mu\delta.[\beta]MS\rfloor^{\mathrm{L}}a$

(gc): Then $P \equiv MS$ and $Q \equiv MS\backslash c$, provided $c \in S$, $c \notin M$, and $\lceil MS\rfloor^{\mathrm{L}}a \triangleq (\nu\overrightarrow{y\alpha})\,(\lceil M\rfloor^{\mathrm{L}}a \mid \lceil\boldsymbol{S}\rfloor^{\mathrm{L}}) \triangleq (\nu\overrightarrow{y\alpha})\,(\lceil M\rfloor^{\mathrm{L}}a \mid \lceil S_c\rfloor^{\mathrm{L}} \mid \lceil S\backslash c\rfloor^{\mathrm{L}}) \equiv (\nu\overrightarrow{y\alpha})\,(\lceil M\rfloor^{\mathrm{L}}a \mid \lceil S\backslash c\rfloor^{\mathrm{L}}) \mid (\nu c)\,\lceil S_c\rfloor^{\mathrm{L}} \approx_{\mathrm{G}} (\nu\overrightarrow{y\alpha})\,(\lceil M\rfloor^{\mathrm{L}}a \mid \lceil S\backslash c\rfloor^{\mathrm{L}}) \triangleq \lceil MS\backslash c\rfloor^{\mathrm{L}}a$

Remember that $\lceil S_c\rfloor^{\mathrm{L}}$ is either $!\,c(v,d).(\lceil v:=N\rfloor^{\mathrm{L}} \mid !\,d\rightarrow\overline{\gamma})$ or $!\,\overline{c}(w).\lceil N\rfloor^{\mathrm{L}}w$ so $\lceil S_c\rfloor^{\mathrm{L}}$ can only input or only output on c, so $(\nu c)\,\lceil S_c\rfloor^{\mathrm{L}} \approx \mathbf{0}$.

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

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

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

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

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

□

Notice that, in the inductive cases, we do not have to deal with processes under guard, so do not need the full power of '≈', as is needed for example for Theorem 7.6 and 7.7, or as Milner and Sangiorgi needed when modelling implicit substitution (see Section 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 Theorem 7.1, the reduction rules (R) and (λS) are modelled using '≡', and that the rules (μ_r), (C) and (nS) are dealt with by the interpretation directly. That leaves five rules where '≈' 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 '$\rightarrow_{\mathrm{XH}}$') *i) If $M \rightarrow^{*}_{\mathrm{XH}} N$, then $\ulcorner M \lrcorner^{\mathrm{L}} a \rightarrow^{*}_{\pi}, \approx_{\mathrm{RGD}} \ulcorner N \lrcorner^{\mathrm{L}} a$. ii) If $M \Uparrow_{\mathrm{XH}}$ then $\ulcorner M \lrcorner^{\mathrm{L}} a \Uparrow_{\pi}$.*

Proof: The first is shown by induction on the length of the reduction sequence, using Theorem 7.1. The second follows from the fact β-reduction is implemented in π by at least one synchronisation, as shown in Example 6.8, and that μ-reduction terminates [49], as does explicit substitution, so non-termination is caused only by β-reduction. □

By Property 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 '$\rightarrow^{*}_{\pi}, \approx_{\mathrm{RGD}}$'. This implies that the following is immediate:

Corollary 7.3 If $M \rightarrow^{*}_{\mathrm{XH}} N$, then $\ulcorner M \lrcorner^{\mathrm{L}} a \approx \ulcorner N \lrcorner^{\mathrm{L}} a$.

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

$$\ulcorner \lambda x.yx \lrcorner^{\mathrm{L}} a \;\triangleq\; (\nu xb)\,((\nu c)\,(\ulcorner y \lrcorner^{\mathrm{L}} c \mid {!}\,c(v,d).(\ulcorner v := x \lrcorner^{\mathrm{L}} \mid {!}\,d \rightarrow \overline{b})) \mid \overline{a}\langle x,b\rangle)$$

is not weakly bisimilar to $\ulcorner y \lrcorner^{\mathrm{L}} a$, and neither is

$$\ulcorner \mu\alpha.[\beta \lrcorner^{\mathrm{L}} y] a \;\triangleq\; \ulcorner y \lrcorner^{\mathrm{L}} \beta\,\{a/\alpha\} \;=\; \ulcorner y \lrcorner^{\mathrm{L}} \beta$$

weakly bisimilar to:

$$\begin{array}{rcl} \ulcorner \lambda x.\mu\gamma.[\beta]y\,\{x\cdot\gamma/\alpha\} \lrcorner^{\mathrm{L}} a &=& \ulcorner \lambda x.\mu\gamma.[\beta]y \lrcorner^{\mathrm{L}} a \qquad\qquad\quad \triangleq \;(\nu xb)\,(\ulcorner \mu\gamma.[\beta]y \lrcorner^{\mathrm{L}} b \mid \overline{a}\langle x,b\rangle) \\ &\triangleq& (\nu xb)\,(\ulcorner y \lrcorner^{\mathrm{L}} \beta\,\{b/\gamma\} \mid \overline{a}\langle x,b\rangle) \;\triangleq\; (\nu xb)\,(\ulcorner y \lrcorner^{\mathrm{L}} \beta \mid \overline{a}\langle x,b\rangle) \end{array}$$

Remember that we have

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

and in the process

$$\begin{array}{rcl} \ulcorner (\lambda x.(\lambda y.M)\,N)\,L \lrcorner^{\mathrm{L}} a &\triangleq& (\nu e)\,((\nu xb)\,((\nu c)\,((\nu yb)\,(\ulcorner M \lrcorner^{\mathrm{L}} b \mid \overline{c}\langle y,b\rangle) \mid \\ && {!}\,c(v,d).(\ulcorner v := N \lrcorner^{\mathrm{L}} \mid {!}\,d \rightarrow \overline{b})) \mid \overline{e}\langle x,b\rangle) \mid {!}\,e(v,d).(\ulcorner v := L \lrcorner^{\mathrm{L}} \mid {!}\,d \rightarrow \overline{a})) \end{array}$$

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 '$\rightarrow_{\mathrm{XH}}$' is not deterministic, and therefore neither is reduction in the image of $\ulcorner \cdot \lrcorner^{\mathrm{L}} \cdot$.

We can make the following observations:

Remark 7.4
- As can be seen from the proofs of Lemma 6.6 and Theorem 7.1, the synchronisations generated by the encoding only involve processes of the shape:

$$x(u).{!}\,u \rightarrow \overline{a} \mid \overline{x}(w).\ulcorner N \lrcorner^{\mathrm{L}} w \quad w(v).\overline{a}\,v \mid (\nu yb)\,(\ulcorner N \lrcorner^{\mathrm{L}} b \mid \overline{w}\langle y,b\rangle) \quad \overline{c}\langle x,b\rangle \mid c(v,d).({!}\,\overline{v}(w).\ulcorner N \lrcorner^{\mathrm{L}} w \mid {!}\,d \rightarrow \overline{a})$$

  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 $y(u).{!}\,u \rightarrow \overline{a} \triangleq \ulcorner y \lrcorner^{\mathrm{L}} a$ that does not occur under guard. All other occurrences of y will appear inside $\ulcorner N \lrcorner^{\mathrm{L}} w$ in subprocesses like ${!}\,c(v,d).({!}\,\overline{v}(w).\ulcorner N \lrcorner^{\mathrm{L}} w \mid {!}\,d \rightarrow \overline{c})$ or $\overline{x}(w).\ulcorner N \lrcorner^{\mathrm{L}} w$, so in particular appear under guard and are unavailable for synchronisation.

We can also show that no reductions are possible in $\ulcorner M \lrcorner^{\mathrm{L}} a$ but those that correspond to reductions in M itself.

Theorem 7.5 (OPERATIONAL COMPLETENESS FOR '$\rightarrow_{\mathrm{XH}}$') *Let $M \in \lambda\mu$, so M is pure.*

i) If $\ulcorner M \rrbracket^{\mathrm{L}} a \rightarrow_{\pi} \mathsf{P}$, then there exists Q, R, and N such that $\mathsf{P} \approx_{\mathrm{R}} \mathsf{Q}$, $\mathsf{Q} \approx_{\mathrm{G}} \mathsf{R}$, and $\mathsf{R} \approx_{\mathrm{D}} \ulcorner N \rrbracket^{\mathrm{L}} a$ (so $\ulcorner M \rrbracket^{\mathrm{L}} a \rightarrow^{+}_{\pi}, \approx_{\mathrm{RGD}} \ulcorner N \rrbracket^{\mathrm{L}} a$) and $M \rightarrow_{\mathrm{XH}} N$.

ii) If $\ulcorner M \rrbracket^{\mathrm{L}} a \rightarrow^{+}_{\pi}, \approx_{\mathrm{RGD}} \ulcorner N \rrbracket^{\mathrm{L}} a$ then $M \rightarrow^{+}_{\mathrm{XH}} N$.

Proof: *i*) By inspection of the cases of the proof for Theorem 7.1, if $\ulcorner M \rrbracket^{\mathrm{L}} a \rightarrow_{\pi} \mathsf{P}$, then there are only two cases where the reduction takes place in the interpreted term directly, and either:

$\ulcorner M \rrbracket^{\mathrm{L}} a = (\nu c)\,((\nu xb)\,(\ulcorner P \rrbracket^{\mathrm{L}} b \mid \overline{c}\langle x,b\rangle) \mid !\,c(v,d).(\ulcorner v := Q \rrbracket^{\mathrm{L}} \mid !\,d \rightarrow \overline{a})) = \ulcorner (\lambda x.P)\,Q \rrbracket^{\mathrm{L}} a$: Then

$$\begin{array}{rll}
\multicolumn{3}{l}{(\nu c)\,((\nu xb)\,(\ulcorner P \rrbracket^{\mathrm{L}} b \mid \overline{c}\langle x,b\rangle) \mid !\,c(v,d).(\ulcorner v := Q \rrbracket^{\mathrm{L}} \mid !\,d \rightarrow \overline{a}))} \\
\rightarrow_{\pi} (c) & (\nu cxb)\,(\ulcorner P \rrbracket^{\mathrm{L}} b \mid \ulcorner x := Q \rrbracket^{\mathrm{L}} \mid !\,b \rightarrow \overline{a} \mid !\,c(v,d).(\ulcorner v := Q \rrbracket^{\mathrm{L}} \mid !\,d \rightarrow \overline{a})) & \\
\equiv & (\nu x)\,((\nu b)\,(\ulcorner P \rrbracket^{\mathrm{L}} b \mid !\,b \rightarrow \overline{a}) \mid \ulcorner x := Q \rrbracket^{\mathrm{L}}) \mid (\nu c)\,(!\,c(v,d).(\ulcorner v := Q \rrbracket^{\mathrm{L}} \mid !\,d \rightarrow \overline{a})) & \\
\approx_{\mathrm{R}} & (\nu x)\,(\ulcorner P \rrbracket^{\mathrm{L}} a \mid \ulcorner x := Q \rrbracket^{\mathrm{L}}) \mid (\nu c)\,(!\,c(v,d).(\ulcorner v := Q \rrbracket^{\mathrm{L}} \mid !\,d \rightarrow \overline{a})) & \\
\approx_{\mathrm{G}} & (\nu x)\,(\ulcorner P \rrbracket^{\mathrm{L}} a \mid \ulcorner x := Q \rrbracket^{\mathrm{L}}) & \triangleq\ \ulcorner P \langle x := Q \rangle \rrbracket^{\mathrm{L}} a
\end{array}$$

Notice that $(\lambda x.P)Q \rightarrow_{\mathrm{XH}} P\langle x := Q\rangle$.

$\ulcorner M \rrbracket^{\mathrm{L}} a = \ulcorner xS_0M_1S_1\cdots M_nS_n \rrbracket^{\mathrm{L}} a$ *with* $\langle x := N\rangle \in S_n$: By Remark 6.5, we can move the substitutions to the outside and then, with $c_{n+1} = a$ and $S = S_0\cdots S_n$:

$$\begin{array}{rll}
\ulcorner xS_0M_1S_1\cdots M_nS_n \rrbracket^{\mathrm{L}} a & \equiv (6.5) & (\nu \overrightarrow{y_n}\, \overrightarrow{\alpha_n})\,(\ulcorner x \rrbracket^{\mathrm{L}} c_1 \mid \overrightarrow{\ulcorner c_i := M_i{\cdot}c_{i+1} \rrbracket^{\mathrm{L}}} \mid \ulcorner \boldsymbol{S} \rrbracket^{\mathrm{L}}) \\
& \triangleq & (\nu \overrightarrow{y_n}\, \overrightarrow{\alpha_n})\,(x(u).!\,u \rightarrow \overline{c_1} \mid \overrightarrow{\ulcorner c_i := M_i{\cdot}c_{i+1} \rrbracket^{\mathrm{L}}} \mid \ulcorner \boldsymbol{S} \rrbracket^{\mathrm{L}}) \\
& \equiv & (\nu \overrightarrow{y_n}\, \overrightarrow{\alpha_n})\,(x(u).!\,u \rightarrow \overline{c_1} \mid \overrightarrow{\ulcorner c_i := M_i{\cdot}c_{i+1} \rrbracket^{\mathrm{L}}} \mid \ulcorner x := N \rrbracket^{\mathrm{L}} \mid \ulcorner \boldsymbol{S} \rrbracket^{\mathrm{L}}) \\
& \triangleq & (\nu \overrightarrow{y_n}\, \overrightarrow{\alpha_n})\,(x(u).!\,u \rightarrow \overline{c_1} \mid \overrightarrow{\ulcorner c_i := M_i{\cdot}c_{i+1} \rrbracket^{\mathrm{L}}} \mid \overline{x}(w).\ulcorner N \rrbracket^{\mathrm{L}} w \mid \ulcorner \boldsymbol{S} \rrbracket^{\mathrm{L}}) \\
& \rightarrow_{\pi} (x) & (\nu \overrightarrow{y_n}\, \overrightarrow{\alpha_n})\,((\nu w)\,(\ulcorner N \rrbracket^{\mathrm{L}} w \mid !\,w \rightarrow \overline{c_1}) \mid \overrightarrow{\ulcorner c_i := M_i{\cdot}c_{i+1} \rrbracket^{\mathrm{L}}} \mid \ulcorner \boldsymbol{S} \rrbracket^{\mathrm{L}}) \\
& \approx_{\mathrm{R}} & (\nu \overrightarrow{y_n}\, \overrightarrow{\alpha_n})\,(\ulcorner N \rrbracket^{\mathrm{L}} c_1 \mid \overrightarrow{\ulcorner c_i := M_i{\cdot}c_{i+1} \rrbracket^{\mathrm{L}}} \mid \ulcorner \boldsymbol{S} \rrbracket^{\mathrm{L}}) \\
& \triangleq & \ulcorner NM_1\cdots M_nS \rrbracket^{\mathrm{L}} a \ \equiv\ \ulcorner NS_0M_1S_1\cdots M_nS_n \rrbracket^{\mathrm{L}} a
\end{array}$$

and $xS_0M_1S_1\cdots M_nS_n \rightarrow_{\mathrm{XH}} NS_0M_1S_1\cdots M_nS_n$.

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 $\ulcorner M \rrbracket^{\mathrm{L}} a \rightarrow_{\pi} \mathsf{P}$, then there exists N such that $\ulcorner M \rrbracket^{\mathrm{L}} a \rightarrow^{*}_{\pi}, \approx_{\mathrm{RGD}} \ulcorner N \rrbracket^{\mathrm{L}} a$ and $M \rightarrow^{*}_{\mathrm{XH}} N$. 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. □

We can also show that standard reduction with explicit substitution, '$\rightarrow_{\mathrm{X}}$', 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 $\ulcorner M \rrbracket^{\mathrm{L}} a \approx \ulcorner N \rrbracket^{\mathrm{L}} a$.*

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 Theorem 7.1.

$x\langle x := N\rangle \rightarrow N$: $\ulcorner x\langle x := N\rangle \rrbracket^{\mathrm{L}} a \ \triangleq\ (\nu x)\,(\ulcorner x \rrbracket^{\mathrm{L}} a \mid \ulcorner x := N \rrbracket^{\mathrm{L}}) \ \triangleq\ (\nu x)\,(x(u).!\,u \rightarrow \overline{a} \mid !\,\overline{x}(w).\ulcorner N \rrbracket^{\mathrm{L}} w) \ \rightarrow_{\pi} (x)$
$(\nu w)\,(!\,w \rightarrow \overline{a} \mid \ulcorner N \rrbracket^{\mathrm{L}} w) \mid (\nu x)\,(!\,\overline{x}(w).\ulcorner N \rrbracket^{\mathrm{L}} w) \ \approx_{\mathrm{G}}\ (\nu w)\,(!\,w \rightarrow \overline{a} \mid \ulcorner N \rrbracket^{\mathrm{L}} w) \ \approx_{\mathrm{R}}\ \ulcorner N \rrbracket^{\mathrm{L}} a$

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

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

$$\begin{array}{l}
M \rightarrow N \Rightarrow LM \rightarrow LN:\ \ulcorner LM \lrcorner^{\mathrm{L}} a \quad \triangleq \quad (\nu c)\,(\ulcorner L \lrcorner^{\mathrm{L}} c \mid !\,c(v,d).(!\,\overline{v}(w).\ulcorner M \lrcorner^{\mathrm{L}} w \mid !\,d \rightarrow \overline{a})) \ \approx (IH) \\
\quad (\nu c)\,(\ulcorner L \lrcorner^{\mathrm{L}} c \mid !\,c(v,d).(!\,\overline{v}(w).\ulcorner N \lrcorner^{\mathrm{L}} w \mid !\,d \rightarrow \overline{a})) \ \triangleq \ \ulcorner LN \lrcorner^{\mathrm{L}} a
\end{array}$$

$$\begin{array}{l}
M \rightarrow N \Rightarrow L\langle x := M\rangle \rightarrow L\langle x := N\rangle:\ \ulcorner L\langle x := M\rangle \lrcorner^{\mathrm{L}} a \ \triangleq \ (\nu x)\,(\ulcorner L \lrcorner^{\mathrm{L}} a \mid !\,\overline{x}(w).\ulcorner M \lrcorner^{\mathrm{L}} w) \ \approx (IH) \\
\quad (\nu x)\,(\ulcorner L \lrcorner^{\mathrm{L}} a \mid !\,\overline{x}(w).\ulcorner N \lrcorner^{\mathrm{L}} w) \ \triangleq \ \ulcorner L\langle x := N\rangle \lrcorner^{\mathrm{L}} a
\end{array}$$

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 $\ulcorner M \lrcorner^{\mathrm{L}} a \approx \ulcorner N \lrcorner^{\mathrm{L}} a$.*

Proof: By induction on the definition of '$=_{\beta\mu}$'. The case $M \rightarrow^{*}_{\beta\mu} N$ follows from the fact that then, by Proposition 4.2, also $M \rightarrow^{*}_{\mathrm{x}} N$, so by Theorem 7.6, we have $\ulcorner M \lrcorner^{\mathrm{L}} a \approx \ulcorner N \lrcorner^{\mathrm{L}} a$. The steps to an equivalence relation follow directly from '$\approx$'. □

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 9.7), but are not all related through '$=_{\beta\mu}$'. However, similar to [50, 51], using a notion of weak equivalence we can deal with the reverse part as well and will do so in sections 9 to 12.

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

Lemma 7.8 If $\mathbf{N}$ is a $\rightarrow_{\mathrm{XH}}$-normal form, then $\ulcorner \mathbf{N} \lrcorner^{\mathrm{L}} a$ is irreducible.

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

$$\begin{array}{l}
\mathbf{N} = xM_1S_1\cdots M_nS_n \ (n \geq 0),\ x \notin S_i, \textit{for } i \in \underline{n}:\ \ulcorner xM_1S_1\cdots M_nS_n \lrcorner^{\mathrm{L}} a \ \triangleq, \equiv \ (6.5) \\
\quad (\nu \overrightarrow{y_n}\,\overrightarrow{\alpha_n}\,) \cdots (\nu \overrightarrow{y_1}\,\overrightarrow{\alpha_1}\,)\,(\nu \vec{c}\,)\,(\ulcorner x \lrcorner^{\mathrm{L}} c_1 \mid \ulcorner c_1 := M_1\cdot c_2 \lrcorner^{\mathrm{L}} \mid \cdots \mid \ulcorner c_n := M_n\cdot a \lrcorner^{\mathrm{L}} \mid \ulcorner S_1 \lrcorner^{\mathrm{L}} \mid \cdots \mid \ulcorner S_n \lrcorner^{\mathrm{L}})
\end{array}$$

Since
$$\begin{array}{rcl}
\ulcorner x \lrcorner^{\mathrm{L}} c_1 & \triangleq & x(u).!\,u \rightarrow \overline{c_1} \\
\ulcorner c_i := M_i \cdot c_{i+1} \lrcorner^{\mathrm{L}} & \triangleq & !\,c_i(v,d).(!\,\overline{v}(w).\ulcorner M_i \lrcorner^{\mathrm{L}} w \mid !\,d \rightarrow \overline{c_{i+1}}) \\
\ulcorner y_j := N_j \lrcorner^{\mathrm{L}} \in S_i & \triangleq & \overline{y_j}(w).\ulcorner N_j \lrcorner^{\mathrm{L}} w \\
\ulcorner \alpha_j := P_j \cdot \gamma_j \lrcorner^{\mathrm{L}} \in S_i & \triangleq & !\,\alpha_j(v,d).(!\,\overline{v}(w).\ulcorner P_j \lrcorner^{\mathrm{L}} w \mid !\,d \rightarrow \overline{\gamma_j})
\end{array}$$
all $\ulcorner M_i \lrcorner^{\mathrm{L}} w$, $\ulcorner N_j \lrcorner^{\mathrm{L}} w$, and $\ulcorner P_j \lrcorner^{\mathrm{L}} w$ 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 S_i, also no synchronisation over x is possible. So this process is in normal form.

$\boldsymbol{N} = \lambda x.\boldsymbol{N}'$: Then $\ulcorner \lambda x.\boldsymbol{N}' \lrcorner^{\mathrm{L}} a \triangleq (\nu xb)\,(\ulcorner \boldsymbol{N}' \lrcorner^{\mathrm{L}} b \mid \overline{a}\langle x,b \rangle)$, and, by induction, $\ulcorner \boldsymbol{N}' \lrcorner^{\mathrm{L}} b$ is in normal form; since b is fresh and $a \notin \ulcorner \boldsymbol{N}' \lrcorner^{\mathrm{L}} b$, that process does not input over a, so $\ulcorner \lambda x.\boldsymbol{N}' \lrcorner^{\mathrm{L}} a$ 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 $\ulcorner \mu\alpha.[\beta]\boldsymbol{N}' \lrcorner^{\mathrm{L}} a \triangleq \ulcorner \boldsymbol{N}' \lrcorner^{\mathrm{L}} \beta\,\{a/\alpha\}$; this case follows immediately by induction. □

Notice that $\ulcorner \mu\alpha.[\beta]\mu\gamma.[\delta]\boldsymbol{N} \lrcorner^{\mathrm{L}} a = \ulcorner \boldsymbol{N} \lrcorner^{\mathrm{L}} \delta\,\{\beta/\gamma\}\,\{a/\alpha\}$, which is in normal form, so some reducible $\lambda\mu\mathbf{x}$-terms 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 $M \rightarrow_{\mathrm{XH}}^{nf} N$, then $\ulcorner M \lrcorner^{\mathrm{L}} a \Downarrow_{\pi}$.*

ii) If $M \rightarrow_{\beta\mu}^{hnf} N$, then $\ulcorner M \lrcorner^{\mathrm{L}} a \Downarrow_{\pi}$.

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

ii) By Proposition 1.8, there exists L in HNF such that $M \rightarrow_{\mathrm{H}}^{nf} L$; by Lemma 5.3, there exists $\boldsymbol{N}$ such that $M \rightarrow_{\mathrm{XH}}^{nf} \boldsymbol{N}$; by the previous part, $\ulcorner M \lrcorner^{\mathrm{L}} a \Downarrow_{\pi}$. □

Notice also that this result is stronger than the formulation of the termination result for Milner's interpretation in [51] (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, Theorem 7.9 immediately leads to:

Corollary 7.10 If $M \Downarrow_{\beta\mu}$, then $\ulcorner M \lrcorner^{\mathrm{L}} a \Downarrow_{\pi}$.

8 On renaming

By Theorem 7.1, renaming might be used during the simulation of $\lambda\mu\mathbf{x}$-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 (Theorem 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, as well as that of open terms. As an illustration of this fact, notice that, as shown in Figure 4, we *can* run the π-process $\ulcorner (\lambda x.xx)(\lambda y.y) \lrcorner^{\mathrm{L}} a$ 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.

$$
\begin{array}{rl}
\ulcorner \lambda x.xx)(\lambda y.y\lrcorner^{\mathrm{L}} a \;\triangleq & (\nu c)\,((\nu xb)\,(\ulcorner xx\lrcorner^{\mathrm{L}} b \mid \overline{c}\langle x,b\rangle) \mid {!}\,c(v,d).(\ulcorner v := \lambda y.y\lrcorner^{\mathrm{L}} \mid {!}\,d\rightarrow\overline{a}))\\
\rightarrow_\pi (c) & (\nu x)\,(\ulcorner xx\lrcorner^{\mathrm{L}} b \mid \ulcorner x := \lambda y.y\lrcorner^{\mathrm{L}} \mid b\rightarrow\overline{a} \mid (\nu c)\,({!}\,c(v,d).(\ulcorner v := \lambda y.y\lrcorner^{\mathrm{L}} \mid d\rightarrow\overline{b})))\\
\approx_{\mathrm{G}} & (\nu x)\,(\ulcorner xx\lrcorner^{\mathrm{L}} b \mid \ulcorner x := \lambda y.y\lrcorner^{\mathrm{L}} \mid b\rightarrow\overline{a})\\
\triangleq & (\nu x)\,((\nu c)\,(x(u).{!}\,u\rightarrow\overline{c} \mid {!}\,c(v,d).(\ulcorner v := x\lrcorner^{\mathrm{L}} \mid {!}\,d\rightarrow\overline{b})) \mid {!}\,\overline{x}(w).\ulcorner \lambda y.y\lrcorner^{\mathrm{L}} w \mid b\rightarrow\overline{a})\\
\rightarrow_\pi (x), \triangleq & (\nu xb)\,((\nu c)\,((\nu w)\,((\nu yb_1)\,(\ulcorner y\lrcorner^{\mathrm{L}} b_1 \mid \overline{w}\langle y,b_1\rangle) \mid {!}\,w\rightarrow\overline{c}) \mid {!}\,c(v,d).(\ulcorner v := x\lrcorner^{\mathrm{L}} \mid d\rightarrow\overline{b})) \mid\\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \ulcorner x := \lambda y.y\lrcorner^{\mathrm{L}} \mid b\rightarrow\overline{a})\\
\rightarrow_\pi (w), \approx_{\mathrm{G}} & (\nu xb)\,((\nu c)\,((\nu yb_1)\,(\ulcorner y\lrcorner^{\mathrm{L}} b_1 \mid \overline{c}\langle y,b_1\rangle) \mid {!}\,c(v,d).(\ulcorner v := x\lrcorner^{\mathrm{L}} \mid d\rightarrow\overline{b})) \mid \ulcorner x := \lambda y.y\lrcorner^{\mathrm{L}} \mid b\rightarrow\overline{a})\\
\rightarrow_\pi (c), \triangleq, \approx_{\mathrm{G}} & (\nu x)\,((\nu yb_1)\,(y(u).{!}\,u\rightarrow\overline{b_1} \mid {!}\,y(w).\ulcorner x\lrcorner^{\mathrm{L}} w \mid b_1\rightarrow\overline{a}) \mid {!}\,\overline{x}(w).\ulcorner \lambda y.y\lrcorner^{\mathrm{L}} w)\\
\rightarrow_\pi (y) & (\nu xb)\,((\nu b_1)\,((\nu w)\,(w\rightarrow\overline{b_1} \mid \ulcorner x\lrcorner^{\mathrm{L}} w) \mid (\nu y)\,(\ulcorner y := x\lrcorner^{\mathrm{L}}) \mid b_1\rightarrow\overline{b}) \mid \ulcorner x := \lambda y.y\lrcorner^{\mathrm{L}} \mid b\rightarrow\overline{a})\\
\triangleq, \approx_{\mathrm{G}} & (\nu xb)\,((\nu b_1)\,((\nu w)\,(w\rightarrow\overline{b_1} \mid x(u).{!}\,u\rightarrow\overline{w}) \mid b_1\rightarrow\overline{b}) \mid\\
& \qquad\qquad\qquad\qquad\qquad \overline{x}(w_1).\ulcorner \lambda y.y\lrcorner^{\mathrm{H}} w_1 \mid {!}\,\overline{x}(w).\ulcorner \lambda y.y\lrcorner^{\mathrm{L}} w \mid b\rightarrow\overline{a})\\
\rightarrow_\pi (x), \approx_{\mathrm{G}} & (\nu b)\,((\nu w_1)\,((\nu b_1)\,((\nu w)\,(w\rightarrow\overline{b_1} \mid w_1\rightarrow\overline{w}) \mid b_1\rightarrow\overline{b}) \mid \ulcorner \lambda y.y\lrcorner^{\mathrm{L}} w_1) \mid b\rightarrow\overline{a})\\
\triangleq & (\nu b)\,((\nu w_1)\,((\nu yb_1)\,((\nu w)\,(w\rightarrow\overline{b_1} \mid w_1\rightarrow\overline{w}) \mid b_1\rightarrow\overline{b}) \mid\\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad (\nu yb_2)\,(\ulcorner y\lrcorner^{\mathrm{L}} b_2 \mid \overline{w_1}\langle y,b_2\rangle))) \mid b\rightarrow\overline{a})\\
\rightarrow_\pi (w_1wb_1b), \triangleq & (\nu yb_2)\,(\ulcorner y\lrcorner^{\mathrm{L}} b_2 \mid \overline{a}\langle y,b_2\rangle) \qquad\qquad\qquad\qquad\qquad\qquad \triangleq\; \ulcorner \lambda y.y\lrcorner^{\mathrm{L}} a
\end{array}
$$

Figure 4. Running $\ulcorner (\lambda x.xx)(\lambda y.y)\lrcorner^{\mathrm{L}} a \rightarrow_\pi^* \ulcorner \lambda y.y\lrcorner^{\mathrm{L}} a$ without renaming.

Example 8.1 When modelling head reduction, we cannot do without renaming completely, not even for closed terms;

$$
\begin{array}{rl}
\ulcorner \lambda x.(\lambda y.y)\,x\lrcorner^{\mathrm{L}} a \;\;\triangleq & (\nu xb)\,((\nu c)\,((\nu yb')\,(\ulcorner y\lrcorner^{\mathrm{L}} b' \mid \overline{c}\langle y,b'\rangle) \mid {!}\,c(v,d).(\ulcorner v := x\lrcorner^{\mathrm{L}} \mid {!}\,d\rightarrow\overline{b})) \mid \overline{a}\langle x,b\rangle)\\
\rightarrow_\pi, \approx_{\mathrm{G}} & (\nu xb)\,((\nu yb')\,(\ulcorner y\lrcorner^{\mathrm{L}} b' \mid \ulcorner y := x\lrcorner^{\mathrm{L}} \mid {!}\,b'\rightarrow\overline{b}) \mid \overline{a}\langle x,b\rangle)\\
\triangleq & (\nu xb)\,((\nu yb')\,(y(u).{!}\,u\rightarrow\overline{b'} \mid {!}\,\overline{y}(w).\ulcorner x\lrcorner^{\mathrm{L}} w \mid {!}\,b'\rightarrow\overline{b}) \mid \overline{a}\langle x,b\rangle)\\
\rightarrow_\pi, \approx_{\mathrm{G}} & (\nu xb)\,((\nu b')\,((\nu w)\,({!}\,w\rightarrow\overline{b'} \mid \ulcorner x\lrcorner^{\mathrm{L}} w) \mid {!}\,b'\rightarrow\overline{b}) \mid \overline{a}\langle x,b\rangle)\\
\triangleq & (\nu xb)\,((\nu b')\,((\nu w)\,({!}\,w\rightarrow\overline{b'} \mid x(u).{!}\,u\rightarrow\overline{w}) \mid {!}\,b'\rightarrow\overline{b}) \mid \overline{a}\langle x,b\rangle)
\end{array}
$$

We would like this to reduce to $(\nu xb)\,(x(u).{!}\,u\rightarrow\overline{b} \mid \overline{a}\langle x,b\rangle) \triangleq \ulcorner \lambda x.x\lrcorner^{\mathrm{L}} a$, but it cannot; the last process above is irreducible. We would therefore need renaming for $(\nu w)\,({!}\,w\rightarrow\overline{b'} \mid x(u).{!}\,u\rightarrow\overline{w}) \approx_{\mathrm{R}} x(u).{!}\,u\rightarrow\overline{b'}$, to achieve

$$
\begin{array}{rl}
& (\nu xb)\,((\nu b')\,((\nu w)\,({!}\,w\rightarrow\overline{b'} \mid x(u).{!}\,u\rightarrow\overline{w}) \mid {!}\,b'\rightarrow\overline{b}) \mid \overline{a}\langle x,b\rangle)\\
\approx_{\mathrm{R}} & (\nu xb)\,((\nu b')\,(x(u).{!}\,u\rightarrow\overline{b'} \mid {!}\,b'\rightarrow\overline{b}) \mid \overline{a}\langle x,b\rangle)\\
\approx_{\mathrm{R}} & (\nu xb)\,(x(u).{!}\,u\rightarrow\overline{b} \mid \overline{a}\langle x,b\rangle) \qquad\qquad\qquad \triangleq\; \ulcorner \lambda x.x\lrcorner^{\mathrm{L}} a
\end{array}
$$

However, we can show that we do not need renaming when interpreting a *weak explicit head reduction* to normal form on closed terms. First we define that notion of reduction.

Definition 8.2 We define *weak explicit head reduction* '$\rightarrow_{w\mathrm{XH}}$' as '$\rightarrow_{\mathrm{XH}}$' in Definition 5.1, but remove the rule:

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

We also define weak explicit head-normal forms.

Definition 8.3 (Weak explicit head-normal forms for $\lambda\mu$) *i*) The $\lambda\mu\mathbf{x}$ *weak explicit head normal forms (*wxhnf*)* are defined through the grammar:

$$
\begin{array}{rcll}
\boldsymbol{H}_{w\mathrm{X}} & ::= & \lambda x.M & (M \in \lambda\mu\mathbf{x})\\
& \mid & xM_1S_1\cdots M_nS_n & (n \geq 0,\ \forall i \in \underline{n}\ (x \notin S_i,\ M_i \in \lambda\mu\mathbf{x},\ c \in S_i \Rightarrow c\ \mathit{free\ in}\ xM_1S_1\cdots M_iS_i))\\
& \mid & \mu\alpha.[\beta]\,\boldsymbol{H}_{w\mathrm{X}} & (\alpha \neq \beta\ \mathit{or}\ \alpha \in \boldsymbol{H}_{w\mathrm{X}},\ \mathit{and}\ \boldsymbol{H}_{w\mathrm{X}} \neq \mu\gamma.[\delta]\boldsymbol{H}'_{w\mathrm{X}})
\end{array}
$$

ii) We say that $M \in \lambda\mu\mathbf{x}$ *has a* wxhnf if there exists $\boldsymbol{H}_{w\mathrm{X}}$ such that $M \rightarrow_{w\mathrm{XH}}^* \boldsymbol{H}_{w\mathrm{X}}$.

Notice that we have chosen not to use the moniker 'lazy'; we could have chosen to also eliminate the rule $(\lambda S) : (\lambda y.M)S \to \lambda y.(MS)$, and this would be a natural choice when dealing with lazy reduction. Since in our interpretation abstraction is modelled using an asynchronous output, however, the step λS is modelled regardless:

$$\begin{array}{lllll}
\lceil (\lambda y.M)S \rfloor^{\mathsf{L}} a & \stackrel{\Delta}{=} & (\nu \vec{x}\,)\,(\lceil \lambda y.M \rfloor^{\mathsf{L}} a \mid \lceil \boldsymbol{S} \rfloor^{\mathsf{L}}) & \stackrel{\Delta}{=} & (\nu \vec{x}\,)\,((\nu yb)\,(\lceil M \rfloor^{\mathsf{L}} b \mid \overline{a}\langle y,b\rangle) \mid \lceil \boldsymbol{S} \rfloor^{\mathsf{L}}) \;\; \equiv \\
 & & (\nu yb)\,((\nu \vec{x}\,)\,(\lceil M \rfloor^{\mathsf{L}} b \mid \lceil \boldsymbol{S} \rfloor^{\mathsf{L}}) \mid \overline{a}\langle x,b\rangle) & \stackrel{\Delta}{=} & (\nu yb)\,(\lceil MS \rfloor^{\mathsf{L}} b \mid \overline{a}\langle x,b\rangle) \;\; \stackrel{\Delta}{=} \\
 & & \lceil \lambda y.(MS) \rfloor^{\mathsf{L}} a & &
\end{array}$$

so allowing it does not alter the structure of the proofs much, apart from the fact that then we would have $(\lambda x.M)S$ 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. We will present a notion of explicit lazy reduction in Definition 13.2.

We can now show:

Theorem 8.4 ($\lceil \cdot \rfloor^{\mathsf{L}}$ PRESERVES '$\to_{w\mathrm{XH}}$' WITHOUT RENAMING) *If P is a closed $\lambda\mu\mathbf{x}$-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 $\lceil P \rfloor^{\mathsf{L}} a \to_{\pi}^{nf} \mathsf{P}$ and $\mathsf{P} \approx_{\mathrm{G}}, \approx_{\mathrm{D}} \lceil Q \rfloor^{\mathsf{L}} a$.*

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

$P = (\lambda x.M)NS_1M_2S_2 \cdots M_nS_n$: Let $S = S_1 \ldots S_n$.

$$\begin{array}{ll}
\lceil P \rfloor^{\mathsf{L}} c_{n+1} \;\stackrel{\Delta}{=},\equiv (6.5)\; (\nu \overrightarrow{y\alpha c}\,)\,(\lceil \lambda x.M \rfloor^{\mathsf{L}} c_1 \mid \overrightarrow{\lceil c_i := M_i \cdot c_{i+1} \rfloor^{\mathsf{L}}} \mid \lceil \boldsymbol{S} \rfloor^{\mathsf{L}}) & \equiv \\
(\nu \overrightarrow{y\alpha c}\,)\,((\nu bx)\,(\lceil M \rfloor^{\mathsf{L}} b \mid \overline{c_1}\langle x,b\rangle) \mid !\,c_1(v,d).(\lceil v := N \rfloor^{\mathsf{L}} \mid !\,d{\to}\overline{c_2}) \mid \overrightarrow{\lceil c_i := M_i \cdot c_{i+1} \rfloor^{\mathsf{L}}} \mid \lceil \boldsymbol{S} \rfloor^{\mathsf{L}}) & \to_{\pi} (c_1) \\
(\nu \overrightarrow{y\alpha c}\,)\,((\nu bx)\,(\lceil M \rfloor^{\mathsf{L}} b \mid \lceil x := N \rfloor^{\mathsf{L}} \mid !\,b{\to}\overline{c_2}) \mid (\nu c_1)\,(\lceil c_1 := N \cdot c_2 \rfloor^{\mathsf{L}}) \mid \overrightarrow{\lceil c_i := M_i \cdot c_{i+1} \rfloor^{\mathsf{L}}} \mid \lceil \boldsymbol{S} \rfloor^{\mathsf{L}}) &
\end{array}$$

At this point the proof of Theorem 7.1 uses renaming to obtain $\lceil M \rfloor^{\mathsf{L}} c_2$, since it might be that M evaluates to a variable y. All synchronisations in the later process take place inside $\lceil M \rfloor^{\mathsf{L}} b$, until that process outputs (on b). By Theorem 7.5, we know that $\lceil M \rfloor^{\mathsf{L}} b$ will run to P such that $\mathsf{P} \approx_{\mathrm{RGD}} \lceil N \rfloor^{\mathsf{L}} b$ with $M \to_{\mathrm{XH}}^{nf} N$.

Since $P \to_{w\mathrm{XH}}^{nf} Q$ and 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 starts with a context switch.

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

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

where G is garbage. So for this case 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.

$P = xS_1M_1 \cdots S_nM_nS_{n+1}$, *with* $\langle x := N\rangle \in S_k$, *for some* $1 \leq k \leq n+1$:

$$\begin{array}{lll}
\lceil xS_1M_1 \cdots S_nM_nS_{n+1} \rfloor^{\mathsf{L}} c_{n+1} & \equiv & (6.5) \\
(\nu \overrightarrow{y\alpha}\,)\,(\lceil x \rfloor^{\mathsf{L}} c_1 \mid \overrightarrow{\lceil c_i := M_i \cdot c_{i+1} \rfloor^{\mathsf{L}}} \mid \lceil \boldsymbol{S} \rfloor^{\mathsf{L}}) & \stackrel{\Delta}{=},\equiv & (!\,\overline{x}(w).\lceil N \rfloor^{\mathsf{L}} w \in \lceil \boldsymbol{S} \rfloor^{\mathsf{L}}) \\
(\nu \overrightarrow{y\alpha}\,)\,(x(u).!\,u{\to}\overline{c_1} \mid \overrightarrow{\lceil c_i := M_i \cdot c_{i+1} \rfloor^{\mathsf{L}}} \mid \overline{x}(w).\lceil N \rfloor^{\mathsf{L}} w \mid \lceil \boldsymbol{S} \rfloor^{\mathsf{L}}) & \to_{\pi} (x) & \\
(\nu \overrightarrow{y\alpha}\,)\,((\nu w)\,(\lceil N \rfloor^{\mathsf{L}} w \mid !\,w{\to}\overline{c_1}) \mid \overrightarrow{\lceil c_i := M_i \cdot c_{i+1} \rfloor^{\mathsf{L}}} \mid \lceil \boldsymbol{S} \rfloor^{\mathsf{L}}) & &
\end{array}$$

Also at this point the proof of Theorem 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. □

In conclusion, weak explicit head reduction on a closed $\lambda\mu$x-term P either generates an

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

Figure 5. Running $\lceil (\lambda x.xx)(\lambda x.xx) \rfloor^{\mathrm{L}} a$

(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.

9 Weak reduction for $\lambda\mu$ and $\lambda\mu\mathbf{x}$

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 $\mathit{0}$, 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 $\mathit{0}$. 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* [56, 57]). 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 9.1 Consider the reduction of $\Delta\Delta$ that was given in Example 5.4; by Theorem 7.1, we have that $\lceil \Delta\Delta \rfloor^{\mathrm{L}} a \approx \lceil zz\,\langle z := y \rangle\,\langle y := x \rangle\,\langle x := \Delta \rangle \rfloor^{\mathrm{L}} a$ as shown in Figure 5, which shows that the interpretation of $\Delta\Delta$ reduces without creating output over a, since that name always occurs inside a sub-process of the shape

$$\lceil c := y \cdot a \rfloor^{\mathrm{L}} \;\stackrel{\Delta}{=}\; {!}\,c(v,d).(\lceil v := y \rfloor^{\mathrm{L}} \mid {!}\,d \rightarrow \overline{a})$$

and does not input, since all occurrences of variables are always bound. So $\lceil \Delta\Delta \rfloor^{\mathrm{L}} a$ is weakly bisimilar to $\mathit{0}$ (see also Lemma 9.7). Therefore,

$$\begin{array}{rcl} \ulcorner\lambda z.\Delta\Delta\lrcorner^{\textsf{L}}\, a & \stackrel{\Delta}{=} & (\nu zb)\,(\ulcorner\Delta\Delta\lrcorner^{\textsf{L}}\, b \mid \overline{a}\langle z,b\rangle) \quad \approx \quad (\nu zb)\,(\mathbf{0} \mid \overline{a}\langle z,b\rangle) \;\approx \\ & & (\nu zb)\,(\ulcorner\Omega\Omega\lrcorner^{\textsf{L}}\, b \mid \overline{a}\langle z,b\rangle) \;\stackrel{\Delta}{=}\; \ulcorner\lambda z.\Omega\Omega\lrcorner^{\textsf{L}}\, a \end{array}$$

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 that equate all unsolvable terms.

We will now introduce the correct notions in $\lambda\mu$.

Definition 9.2 As in Definition 8.2, we define the notion '$\rightarrow_{w\beta\mu}$' of *weak $\beta\mu$-reduction* as in Definition 1.4, the notion '$\rightarrow_{w\textsf{H}}$' of *weak head reduction* on $\lambda\mu$ as in Definition 1.7 by (also) eliminating the contextual rule:

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

We have already defined the notion *weak explicit head-reduction* '$\rightarrow_{w\textsf{XH}}$' on $\lambda\mu\mathbf{x}$ in Definition 8.2.

We can show the following property.

Lemma 9.3 i) Let $M,N \in \lambda\mu$; then $M \rightarrow^{nf}_{w\textsf{H}} N$ if and only if there exists $N' \in \lambda\mu\mathbf{x}$ such that $M \rightarrow^{nf}_{w\textsf{XH}} N'$, and $N' \rightarrow^{nf}_{:=} N$.

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

Proof: Straightforward, similar to Lemma 5.3. □

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

Definition 9.4 (Weak head-normal forms for $\lambda\mu$) *i*) The $\lambda\mu$ *weak head-normal forms* (whnf) are defined through the grammar:

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

ii) We say that M *has a* whnf if there exists $\boldsymbol{H}_w$ such that $M \rightarrow^{*}_{w\textsf{H}} \boldsymbol{H}_w$.

As before, it is easy to verify that whnfs are the 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 '$\rightarrow_{w\textsf{XH}}$' $\subseteq$ '$\rightarrow_{\textsf{XH}}$', the equivalent of Proposition 1.8 and Theorem 7.2 also hold for *weak explicit head reduction*.

Proposition 9.5 If $M \rightarrow^{*}_{\beta\mu} N$ with N in whnf*, then there exists $\boldsymbol{H}_w$ such that $M \rightarrow^{nf}_{w\textsf{H}} \boldsymbol{H}_w$ and $\boldsymbol{H}_w \rightarrow^{*}_{\beta\mu} N$ without using '$\rightarrow_{w\textsf{H}}$'.*

Theorem 9.6 *i) If $M \rightarrow^{*}_{w\textsf{XH}} N$, then $\ulcorner M\lrcorner^{\textsf{L}}\, a \approx \ulcorner N\lrcorner^{\textsf{L}}\, a$.*

ii) If $M \Uparrow_{w\textsf{XH}}$, then $\ulcorner M\lrcorner^{\textsf{L}}\, a \Uparrow_{\pi}$.

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

Lemma 9.7 If M has no wxhnf *(so M also has no* whnf*), then $\ulcorner M\lrcorner^{\textsf{L}}\, a \approx \mathbf{0}$.*

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 $\mathit{0}$.

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 $\ulcorner N \lrcorner^{\mathrm{L}} \beta\,\{a/\alpha\} \approx \mathit{0}$, then also $\ulcorner M \lrcorner^{\mathrm{L}} a \approx \mathit{0}$; therefore we can assume M itself does not start with a context switch.

Let $M = RS_0P_1S_1\cdots P_nS_n$ with $n \geq 0$ and each S_i possibly empty, and let $S = S_0S_1\cdots S_n$, and:

$$\begin{array}{rcl}
\ulcorner M \lrcorner^{\mathrm{L}} a & \equiv(6.5) & (\nu \overrightarrow{\alpha y}\,)\,((\nu \vec{c}\,)\,(\ulcorner R \lrcorner^{\mathrm{L}} c_1 \mid \overrightarrow{\ulcorner c_i := P_i{\cdot}c_{i+1} \lrcorner^{\mathrm{L}}}\,) \mid \ulcorner \boldsymbol{S} \lrcorner^{\mathrm{L}}) \\
\ulcorner c_i := P_i{\cdot}c_{i+1} \lrcorner^{\mathrm{L}} & \triangleq & !\,c_i(v,d).(!\,\overline{v}(w).\ulcorner P_i \lrcorner^{\mathrm{L}} w \mid !\,d \rightarrow \overline{c_{i+1}}) \\
\ulcorner \boldsymbol{S} \lrcorner^{\mathrm{L}} & \equiv & \overrightarrow{!\,\overline{y_j}(w).\ulcorner Q_j \lrcorner^{\mathrm{L}} w} \mid \overrightarrow{!\,\alpha_k(v,d).(\ulcorner v := T_k \lrcorner^{\mathrm{L}} \mid !\,d \rightarrow \overline{\beta_k})}
\end{array}$$

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 $\ulcorner R \lrcorner^{\mathrm{L}} c_1$. We reason by coinduction 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 $K\langle x := P_1\rangle S_2P_2S_1\cdots P_nS_n$. By coinduction, the process

$$\begin{array}{l}
\ulcorner K\langle x := P_1\rangle S_2P_2S_1\cdots P_nS_n \lrcorner^{\mathrm{L}} a \ \triangleq,\equiv \\
\qquad (\nu \overrightarrow{\alpha y}\,)\,((\nu \vec{c}\,)\,((\nu x)\,(\ulcorner K \lrcorner^{\mathrm{L}} c_1 \mid !\,\overline{x}(w).\ulcorner P_1 \lrcorner^{\mathrm{L}} w) \mid \overrightarrow{\ulcorner c_i := P_i{\cdot}c_{i+1} \lrcorner^{\mathrm{L}}}\,) \mid \ulcorner \boldsymbol{S} \lrcorner^{\mathrm{L}})
\end{array}$$

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

(μ_p): Then $R = \mu\alpha.[\alpha]K$, and $n \geq 1$, and M contracts to $(\mu\gamma.[\gamma]K\langle \alpha := P_1{\cdot}\gamma\rangle P_1)S_2P_2S_1\cdots P_nS_n$. By coinduction, the process

$$\begin{array}{rl}
\multicolumn{2}{l}{\ulcorner (\mu\gamma.[\gamma]K\langle \alpha := P_1{\cdot}\gamma\rangle P_1)S_2P_2S_1\cdots P_nS_n \lrcorner^{\mathrm{L}} a} \\
\triangleq,\equiv & (\nu \overrightarrow{\alpha y}\,)\,((\nu \vec{c}\,)\,((\nu c')\,((\nu\alpha)\,(\ulcorner K \lrcorner^{\mathrm{L}} c' \mid \ulcorner \alpha := P_1{\cdot}\gamma \lrcorner^{\mathrm{L}}) \mid \ulcorner c' := P_1{\cdot}\gamma \lrcorner^{\mathrm{L}})\,\{c_1/\gamma\} \mid \overrightarrow{\ulcorner c_i := P_i{\cdot}c_{i+1} \lrcorner^{\mathrm{L}}}\,) \mid \ulcorner \boldsymbol{S} \lrcorner^{\mathrm{L}}) \\
= & (\nu \overrightarrow{\alpha y}\,)\,((\nu \vec{c}\,)\,((\nu c')\,((\nu\alpha)\,(\ulcorner K \lrcorner^{\mathrm{L}} c' \mid \ulcorner \alpha := P_1{\cdot}c_1 \lrcorner^{\mathrm{L}}) \mid \ulcorner c' := P_1{\cdot}c_1 \lrcorner^{\mathrm{L}}) \mid \overrightarrow{\ulcorner c_i := P_i{\cdot}c_{i+1} \lrcorner^{\mathrm{L}}}\,) \mid \ulcorner \boldsymbol{S} \lrcorner^{\mathrm{L}})
\end{array}$$

does not exhibit inputs or outputs, so, in particular, $(\nu \overrightarrow{\alpha y}\ \vec{c}\ c')\,(\ulcorner K \lrcorner^{\mathrm{L}} c'\,\{c_1/\gamma\})$ (the only one part that could) does not, and thereby neither does $(\nu \overrightarrow{\alpha y}\ \vec{c}\ c')\,(\ulcorner K \lrcorner^{\mathrm{L}} c')$, so neither does $(\nu \overrightarrow{\alpha y}\ \vec{c}\ c')\,(\ulcorner K \lrcorner^{\mathrm{L}} \alpha)$ so also

$$\begin{array}{rl}
\multicolumn{2}{l}{(\nu \overrightarrow{\alpha y}\,)\,((\nu \vec{c}\,)\,(\ulcorner K \lrcorner^{\mathrm{L}} \alpha \mid \ulcorner \alpha := P_1{\cdot}c_2 \lrcorner^{\mathrm{L}} \mid \overrightarrow{\ulcorner c_i := P_i{\cdot}c_{i+1} \lrcorner^{\mathrm{L}}}\,) \mid \ulcorner \boldsymbol{S} \lrcorner^{\mathrm{L}})} \\
= & (\nu \overrightarrow{\alpha y}\,)\,((\nu \vec{c}\,)\,(\ulcorner K \lrcorner^{\mathrm{L}} \alpha\,\{c_1/\alpha\} \mid \ulcorner c_1 := P_1{\cdot}c_2 \lrcorner^{\mathrm{L}} \mid \overrightarrow{\ulcorner c_i := P_i{\cdot}c_{i+1} \lrcorner^{\mathrm{L}}}\,) \mid \ulcorner \boldsymbol{S} \lrcorner^{\mathrm{L}}) \\
\approx_{\mathrm{D}} & (\nu \overrightarrow{\alpha y}\,)\,((\nu \vec{c}\,)\,(\ulcorner K \lrcorner^{\mathrm{L}} \alpha\,\{c_1/\alpha\} \mid \overrightarrow{\ulcorner c_i := P_i{\cdot}c_{i+1} \lrcorner^{\mathrm{L}}}\,) \mid \ulcorner \boldsymbol{S} \lrcorner^{\mathrm{L}}) \\
\triangleq & \ulcorner (\mu\alpha.[\alpha]K)P_1S_2P_2S_1\cdots P_nS_n \lrcorner^{\mathrm{L}} a
\end{array}$$

does not.

(μ_r): Then $R = \mu\alpha.[\beta]K$, and M contracts to $(\mu\gamma.[\beta]K\langle \alpha := P_1{\cdot}\gamma\rangle)S_2P_2S_1\cdots P_nS_n$. Since, as in the proof of Theorem 7.1,

$$\begin{array}{lr}
\ulcorner (\mu\alpha.[\beta]K)P_1S_2P_2S_1\cdots P_nS_n \lrcorner^{\mathrm{L}} a & \triangleq,\equiv \\
(\nu \overrightarrow{\alpha y}\,)\,((\nu \vec{c}\,)\,((\nu c_1)\,(\ulcorner \mu\alpha.[\beta]K \lrcorner^{\mathrm{L}} c_1 \mid \ulcorner c_1 := P_1{\cdot}c_2 \lrcorner^{\mathrm{L}}) \mid \overrightarrow{\ulcorner c_i := P_i{\cdot}c_{i+1} \lrcorner^{\mathrm{L}}}\,) \mid \ulcorner \boldsymbol{S} \lrcorner^{\mathrm{L}}) & \triangleq \\
(\nu \overrightarrow{\alpha y}\,)\,((\nu \vec{c}\,)\,(\ulcorner K \lrcorner^{\mathrm{L}} \beta\,\{c_1/\alpha\} \mid \ulcorner c_1 := P_1{\cdot}c_2 \lrcorner^{\mathrm{L}} \mid \overrightarrow{\ulcorner c_i := P_i{\cdot}c_{i+1} \lrcorner^{\mathrm{L}}}\,) \mid \ulcorner \boldsymbol{S} \lrcorner^{\mathrm{L}}) & =_{\alpha} \\
(\nu \overrightarrow{\alpha y}\,)\,((\nu \vec{c}\,)\,(\ulcorner K \lrcorner^{\mathrm{L}} \beta \mid \ulcorner \alpha := P_1{\cdot}c_2 \lrcorner^{\mathrm{L}} \mid \overrightarrow{\ulcorner c_i := P_i{\cdot}c_{i+1} \lrcorner^{\mathrm{L}}}\,) \mid \ulcorner \boldsymbol{S} \lrcorner^{\mathrm{L}}) & \triangleq \\
(\nu \overrightarrow{\alpha y}\,)\,((\nu \vec{c}\,)\,((\nu\alpha)\,(\ulcorner K \lrcorner^{\mathrm{L}} \beta \mid \ulcorner \alpha := P_1{\cdot}\gamma \lrcorner^{\mathrm{L}})\,\{c_2/\gamma\} \mid \overrightarrow{\ulcorner c_i := P_i{\cdot}c_{i+1} \lrcorner^{\mathrm{L}}}\,) \mid \ulcorner \boldsymbol{S} \lrcorner^{\mathrm{L}}) & \triangleq,\equiv \\
\ulcorner (\mu\gamma.[\beta]K\langle \alpha := P_1{\cdot}\gamma\rangle)S_2P_2S_1\cdots P_nS_n \lrcorner^{\mathrm{L}} a &
\end{array}$$

the result follows immediately by co-induction.

(hv): Then $R = x$ and $\langle x := N\rangle \in S$, and M contracts to $NS_0P_1S_1\cdots P_nS_n$. By coinduction,

$$\begin{array}{ll}
\ulcorner\Delta\Delta\lrcorner^{\mathrm{L}} a \quad = \quad \ulcorner(\lambda x.xx)\,\Delta\lrcorner^{\mathrm{L}} a & \triangleq \\
(\nu c)\,((\nu xb)\,(\ulcorner xx\lrcorner^{\mathrm{L}} b \mid \overline{c}\langle x,b\rangle) \mid {!}\,c(v,d).(\ulcorner v := \Delta\lrcorner^{\mathrm{L}} \mid {!}\,d\to\overline{a})) & \to(c) \\
(\nu xb)\,(\ulcorner xx\lrcorner^{\mathrm{L}} b \mid \ulcorner x := \Delta\lrcorner^{\mathrm{L}} \mid {!}\,b\to\overline{a}) \mid (\nu c)\,({!}\,c(v,d).(\ulcorner v := \Delta\lrcorner^{\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).(\ulcorner v := x\lrcorner^{\mathrm{L}} \mid {!}\,d\to\overline{b})) \mid {!}\,\overline{x}(w).\ulcorner\Delta\lrcorner^{\mathrm{L}} w \mid {!}\,b\to\overline{a}) & \to(x) \\
(\nu xbw)\,((\nu c)\,({!}\,w\to\overline{c} \mid {!}\,c(v,d).(\ulcorner v := x\lrcorner^{\mathrm{L}} \mid {!}\,d\to\overline{b})) \mid (\nu yb)\,(\ulcorner yy\lrcorner^{\mathrm{L}} b \mid \overline{w}\langle y,b\rangle) \mid \ulcorner x := \Delta\lrcorner^{\mathrm{L}} \mid {!}\,b\to\overline{a}) & \to(w) \\
(\nu xb)\,((\nu c)\,((\nu yb)\,(\ulcorner yy\lrcorner^{\mathrm{L}} b \mid \overline{c}\langle y,b\rangle) \mid {!}\,c(v,d).(\ulcorner v := x\lrcorner^{\mathrm{L}} \mid {!}\,d\to\overline{b}) \mid \ulcorner x := \Delta\lrcorner^{\mathrm{L}} \mid {!}\,b\to\overline{a}) & \to(c),\approx_{\mathrm{G}} \\
(\nu xb)\,((\nu yb_1)\,(\ulcorner yy\lrcorner^{\mathrm{L}} b_1 \mid \ulcorner y := x\lrcorner^{\mathrm{L}} \mid {!}\,b_1\to\overline{b}) \mid \ulcorner x := \Delta\lrcorner^{\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).(\ulcorner v := y\lrcorner^{\mathrm{L}} \mid {!}\,d\to\overline{b_1})) \mid {!}\,\overline{y}(w).\ulcorner x\lrcorner^{\mathrm{L}} w \mid {!}\,b_1\to\overline{b}) \mid \ulcorner x := \Delta\lrcorner^{\mathrm{L}} \mid {!}\,b\to\overline{a}) & \to(y) \\
(\nu xb)\,((\nu yb_1)\,((\nu c)\,(w\to\overline{c} \mid {!}\,c(v,d).(\ulcorner v := y\lrcorner^{\mathrm{L}} \mid {!}\,d\to\overline{b_1})) \mid \ulcorner x\lrcorner^{\mathrm{L}} w \mid \ulcorner y := x\lrcorner^{\mathrm{L}} \mid {!}\,b_1\to\overline{b}) \mid \ulcorner x := \Delta\lrcorner^{\mathrm{L}} \mid {!}\,b\to\overline{a}) & \triangleq \\
(\nu xb)\,((\nu yb_1)\,((\nu c)\,(w\to\overline{c} \mid {!}\,c(v,d).(\ulcorner v := y\lrcorner^{\mathrm{L}} \mid {!}\,d\to\overline{b_1})) \mid x(u).{!}\,u\to\overline{w} \mid \ulcorner y := x\lrcorner^{\mathrm{L}} \mid {!}\,b_1\to\overline{b}) \mid \ulcorner x := \Delta\lrcorner^{\mathrm{L}} \mid {!}\,b\to\overline{a}) & \equiv \\
(\nu xb)\,((\nu yb_1)\,((\nu c)\,(w\to\overline{c} \mid {!}\,c(v,d).(\ulcorner v := y\lrcorner^{\mathrm{L}} \mid {!}\,d\to\overline{b_1})) \mid x(u).{!}\,u\to\overline{w} \mid \ulcorner y := x\lrcorner^{\mathrm{L}} \mid {!}\,b_1\to\overline{b}) \mid & \\
\qquad\qquad\qquad {!}\,\overline{x}(w).(\nu zb)\,(\ulcorner zz\lrcorner^{\mathrm{L}} b \mid \overline{w}\langle z,b\rangle) \mid \ulcorner x := \Delta\lrcorner^{\mathrm{L}} \mid {!}\,b\to\overline{a}) & \to(x,w_1,w,c) \\
(\nu xb)\,((\nu yb_1)\,((\nu zb_2)\,(\ulcorner zz\lrcorner^{\mathrm{L}} b_2 \mid \ulcorner z := y\lrcorner^{\mathrm{L}} \mid {!}\,b_2\to\overline{b_1})) \mid \ulcorner y := x\lrcorner^{\mathrm{L}} \mid {!}\,b_1\to\overline{b} \mid \ulcorner x := \Delta\lrcorner^{\mathrm{L}} \mid {!}\,b\to\overline{a})) & \dots
\end{array}$$

Figure 6. Running $\ulcorner\Delta\Delta\lrcorner^{\mathrm{L}} a$ without renaming, but using garbage collection.

$$\ulcorner NS_0P_1S_1\cdots P_nS_n\lrcorner^{\mathrm{L}} a \;\triangleq\; (\nu\overrightarrow{\alpha y}\,)\,((\nu\vec{c}\,)\,(\ulcorner N\lrcorner^{\mathrm{L}} c_1 \mid \overrightarrow{\ulcorner c_i := P_i{\cdot}c_{i+1}\lrcorner^{\mathrm{L}}}\,) \mid \ulcorner\boldsymbol{S}\lrcorner^{\mathrm{L}})$$

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

$$\ulcorner xS_1P_1S_2P_2S_1\cdots P_nS_n\lrcorner^{\mathrm{L}} a \;\triangleq,\equiv\; (\nu\overrightarrow{\alpha y}\,)\,((\nu\vec{c}\,)\,(x(u).{!}\,u\to\overline{c_1} \mid \overrightarrow{\ulcorner c_i := P_i{\cdot}c_{i+1}\lrcorner^{\mathrm{L}}}\,) \mid \overline{x}(w).\ulcorner N\lrcorner^{\mathrm{L}} w \mid \ulcorner\boldsymbol{S}\lrcorner^{\mathrm{L}})$$

where $x \in \vec{y}$, does not. □

The reduction of $\ulcorner\Delta\Delta\lrcorner^{\mathrm{L}} a$ is given in Figure 6, and shows that the interpretation of $\Delta\Delta$ reduces without creating output over a; notice that the individual steps of the above reduction in '$\rightarrow_{\mathrm{XH}}$' in Example 5.4 are respected in Figure 6.

As a direct consequence of Lemma 9.7, as for Milner's and Sangiorgi's interpretations, our interpretation is not extensional, since $\ulcorner\Delta\Delta\lrcorner^{\mathrm{L}} a \approx \mathit{0}$, whereas

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

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 9.8 i) If $M \rightarrow_{w\mathrm{XH}}^{nf} \lambda x.N$, then $\ulcorner M\lrcorner^{\mathrm{L}} a \Downarrow \overline{a}$.

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

iii) If $M \rightarrow_{w\mathrm{XH}}^{nf} xN_1S_1\cdots N_nS_n$ or $M \rightarrow_{w\mathrm{XH}}^{nf} \mu\alpha.[\beta]\,xN_1S_1\cdots N_nS_1$, then $\ulcorner M\lrcorner^{\mathrm{L}} a \Downarrow x$.

Proof: Since '$\rightarrow_{w\mathrm{XH}}^{*}$' $\subseteq$ '$\rightarrow_{\mathrm{XH}}^{*}$', this follows from Theorem 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 will now 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 9.9 i) If $\ulcorner M\lrcorner^{\mathrm{L}} a \Downarrow \overline{a}$, then there exist x, $N \in \lambda\mu\mathbf{x}$ such that $\ulcorner M\lrcorner^{\mathrm{L}} a \approx \ulcorner\lambda x.N\lrcorner^{\mathrm{L}} a$, and $M \rightarrow_{w\mathrm{XH}}^{nf} \lambda x.N$.

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

iii) If $\ulcorner M\lrcorner^{\mathrm{L}} a \not\Downarrow_o$ but $\ulcorner M\lrcorner^{\mathrm{L}} a \Downarrow x$, then there exist $N_1,\ldots,N_n$, c, and $S_1,\ldots,S_n$ with $n \geq 0$ such that:

- *$\ulcorner M\lrcorner^{\mathrm{L}} a \approx \ulcorner xN_1S_1\cdots N_nS_n\lrcorner^{\mathrm{L}} c$;*
- *$M \rightarrow_{w\mathrm{XH}}^{nf} xN_1S_1\cdots N_nS_n$ if $a = c$;*

- $M \rightarrow^{nf}_{w\mathrm{XH}} \mu\alpha.[c]\, xN_1S_1 \cdots N_nS_n$, *if* $a \neq c$.

Proof: *i*) By checking the proof for Theorem 7.5, we observe that if $\ulcorner M \lrcorner^{\mathrm{L}} a$ 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 $M \rightarrow^{nf}_{w\mathrm{XH}} \lambda x.N$. Since '$\rightarrow_{w\mathrm{XH}}$' $\subseteq$ '$\rightarrow_{\mathrm{XH}}$', also $M \rightarrow^{*}_{\mathrm{XH}} \lambda x.N$, and by Theorem 7.1 we get $\ulcorner M \lrcorner^{\mathrm{L}} a \approx \ulcorner \lambda x.N \lrcorner^{\mathrm{L}} a$.

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

iii) If $\ulcorner M \lrcorner^{\mathrm{L}} a$ runs to a process that inputs, but does not output, then by the proof for Theorem 7.5 and Remark 7.4 M runs to a term with a head variable and without outermost abstractions, so there exist $N_1,\ldots,N_n$, x, and $S_1,\ldots,S_n$ with $n \geq 0$ and α,β such that either:

$M \rightarrow^{nf}_{w\mathrm{XH}} xN_1S_1 \cdots N_nS_n$: Since also $M \rightarrow^{*}_{\mathrm{XH}} xN_1S_1 \cdots N_nS_n$, we get that $\ulcorner M \lrcorner^{\mathrm{L}} a \approx \ulcorner xN_1S_1 \cdots N_nS_n \lrcorner^{\mathrm{L}} c$ follows by Theorem 7.1.

$M \rightarrow^{nf}_{w\mathrm{XH}} \mu\alpha.[\beta]xN_1S_1 \cdots N_nS_n$: Since also $M \rightarrow^{*}_{\mathrm{XH}} \mu\alpha.[\beta]xN_1S_1 \cdots N_nS_n$, by Theorem 7.1 we get $\ulcorner M \lrcorner^{\mathrm{L}} a \approx \ulcorner \mu\alpha.[\beta \lrcorner^{\mathrm{L}} xN_1S_1 \cdots N_nS_n]a = \ulcorner xN_1S_1 \cdots N_nS_n \lrcorner^{\mathrm{L}} \beta \{a/\alpha\}$. □

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:

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

Notice that $\Delta\Delta \sim_{w\beta\mu} \Omega\Omega$ and $\lambda z.\Delta\Delta \sim_{w\beta\mu} \lambda z.\Omega\Omega$, 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 Property 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 Theorem 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 $M'' \sim_{w\text{H}} N''$.

Notice that $\lambda z.\Delta\Delta \sim_{w\text{H}} \lambda z.\Omega\Omega$ because $\Delta\Delta \sim_{w\text{H}} \Omega\Omega$, 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 Definition 9.4, 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 $\to_{w\text{XH}}$-normal form, or
- both $M \to^{nf}_{w\text{XH}} M'$ and $N \to^{nf}_{w\text{XH}} N'$, and either:
  - if $M' = xM_1S_1 \cdots M_nS_n$ $(n \geq 0)$, then $N' = xN_1S'_1 \cdots N_nS'_n$ (so $x \notin S_i$, $x \notin S'_i$, for $i \in \underline{n}$) and $M_iS \sim_{w\text{XH}} N_iS'$ for all $i \in \underline{n}$ where $S = S_1 \cdots S_n$ and $S' = S'_1 \cdots S'_n$; 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 9.3, there exist $\overrightarrow{M'_i}$, $\overrightarrow{N'_i}$ such that both

$$\begin{array}{lllll} M & \to^{nf}_{w\text{XH}} & xM'_1S_1 \cdots M'_nS_n & \to^{*}_{:=} & xM_1 \cdots M_n \quad \text{and} \\ N & \to^{nf}_{w\text{XH}} & xN'_1S'_1 \cdots N'_nS'_n & \to^{*}_{:=} & xN_1 \cdots N_n \end{array}$$

  Let $S = S_1 \cdots S_n$ and $S' = S'_1 \cdots S'_n$, then in particular $M'_iS \to^{nf}_{:=} M_i$ and $N'_iS' \to^{nf}_{:=} N_i$, for all $i \in \underline{n}$; then by induction, $M'_iS \sim_{w\text{XH}} N'_iS'$ 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 9.3 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

$N' \rightarrow_{:=}^{nf} N$, and $M' \sim_{w\mathrm{XH}} N'$, then either:

- $M' \rightarrow_{w\mathrm{XH}}^{nf} xM'_1S_1 \cdots M'_nS_n$, $N' \rightarrow_{w\mathrm{XH}}^{nf} xN'_1S'_1 \cdots N'_nS'_n$, $S = S_1 \cdots S_n$ and $S' = S'_1 \cdots S'_n$, and $M'_iS \sim_{w\mathrm{XH}} N'_iS'$, for all $i \in \underline{n}$. Let, for all $i \in \underline{n}$, $M'_iS \rightarrow_{:=}^{nf} M_i$ and $N'_iS' \rightarrow_{:=}^{nf} N_i$ then by induction, $M_i \sim_{w\mathrm{H}} N_i$, for all $i \in \underline{n}$. Let $M' \rightarrow_{:=}^{nf} M$; since we have $M' \rightarrow_{w\mathrm{XH}}^{nf} xM'_1S_1 \cdots M'_nS_n \rightarrow_{:=}^{nf} xM_1 \cdots M_n$, by Lemma 9.3, $M \rightarrow_{w\mathrm{H}}^{nf} xM_1 \cdots M_n$. Likewise, we have $N \rightarrow_{w\mathrm{H}}^{nf} xN_1 \cdots N_n$. But then $M \sim_{w\mathrm{H}} N$.
- $M' \rightarrow_{w\mathrm{XH}}^{nf} \lambda x.P'$, $N' \rightarrow_{w\mathrm{XH}}^{nf} \lambda x.Q'$, and $P' \sim_{w\mathrm{XH}} Q'$. Let $P' \rightarrow_{:=}^{nf} P$, and $Q' \rightarrow_{:=}^{nf} Q$, then by induction, $P \sim_{w\mathrm{H}} Q$. Then we have $M' \rightarrow_{w\mathrm{XH}}^{nf} \lambda x.P' \rightarrow_{:=}^{nf} \lambda x.P$, and by Lemma 9.3, $M \rightarrow_{w\mathrm{H}}^{nf} \lambda x.P$. Similarly, we have $N \rightarrow_{w\mathrm{H}}^{nf} \lambda x.Q$; so $M \sim_{w\mathrm{H}} N$.
- $M' \rightarrow_{w\mathrm{XH}}^{nf} \mu\delta.[\gamma]P'$, $N' \rightarrow_{w\mathrm{XH}}^{nf} \mu\delta.[\gamma]Q'$, and $P' \sim_{w\mathrm{XH}} Q'$; 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 $M \sim_{w\mathrm{XH}} N \iff M \sim_{w\mathrm{H}} N$.

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 Theorem 12.1 that $\ulcorner M \lrcorner^{\mathrm{L}} a \approx \ulcorner N \lrcorner^{\mathrm{L}} a \iff M \sim_{w\mathrm{XH}} N$. To complete the proof towards '$\sim_{w\beta\mu}$', we are thus left with the obligation to show that $M \sim_{w\mathrm{XH}} N \iff M \sim_{w\beta\mu} N$. In Corollary 10.6 we have shown that $M \sim_{w\mathrm{XH}} N \iff M \sim_{w\mathrm{H}} N$, for pure terms; to achieve $M \sim_{w\mathrm{H}} N \iff M \sim_{w\beta\mu} N$, in this section we go through a notion of *weak approximation*. Based on Wadsworth's approach [56], we define '$\sim_{\mathcal{A}_w}$' that expresses that terms have the same weak approximants and show that $M \sim_{w\mathrm{H}} N \iff M \sim_{\mathcal{A}_w} N \iff M \sim_{w\beta\mu} N$.

The notions of *approximant* and *approximation* were first introduced by Wadsworth for the λ-calculus [56], 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 $\perp$. 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 $\lambda x_1 \cdots x_m.yN_1 \cdots N_n$ (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 [56] defines $\mathcal{A}(M)$ (similar to Definition 11.1 below) as the set of approximants of the λ-term M, which forms a meet semi-lattice; in [57], the connection is established between approximation and semantics, by showing

$$\ulcorner M \lrcorner_{\mathrm{D}_\infty} p \;=\; \bigsqcup \{\, \ulcorner A \lrcorner_{\mathrm{D}_\infty} p \mid A \in \mathcal{A}(M) \,\}.$$

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 $\perp$; thus, they *approximate* the result of computations. Intuitively, an approximant can be seen as a 'snapshot' of a computation, where we focus on that part of the resulting program which will no longer change, which corresponds to the (observable) *output*.

We now define a *weak approximation semantics* for $\lambda\mu$. Approximation for $\lambda\mu$ has been studied by others as well [53, 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 Definition 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 '$\rightarrow_{\beta\mu}$' is defined through the grammar:[16]

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

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.*:

$$\begin{array}{rcl}\bot & \sqsubseteq & M \\ x & \sqsubseteq & x \\ M \sqsubseteq M' & \Rightarrow & \lambda x.M \sqsubseteq \lambda x.M' \;\&\; \mu\gamma.[\delta]M \sqsubseteq \mu\gamma.[\delta]M' \\ M_1 \sqsubseteq M'_1 \wedge M_2 \sqsubseteq M'_2 & \Rightarrow & M_1M_2 \sqsubseteq M'_1M'_2\end{array}$$

iv) The set of *weak approximants* of $M \in \lambda\mu$, $\mathcal{A}_w(M)$, is defined through:[17]

$$\mathcal{A}_w(M) \;\triangleq\; \{\,\boldsymbol{A}_w \in \mathcal{A}_w \mid \exists N \in \lambda\mu \;(M \rightarrow^*_{\beta\mu} N \wedge \boldsymbol{A}_w \sqsubseteq N)\,\}.$$

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 $\boldsymbol{A}_1 \sqsubseteq M_1$ and $\boldsymbol{A}_2 \sqsubseteq M_2$, then $\boldsymbol{A}_1\boldsymbol{A}_2$ need not be an approximant; it is one if $\boldsymbol{A}_1 = x\boldsymbol{A}_1^1\cdots\boldsymbol{A}_1^n$, 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 '$\rightarrow_{\beta\mu}$' by adding the reduction rules:

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

(so not $\lambda x.\bot \rightarrow \bot$) 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 $\boldsymbol{A}_w \sqsubseteq M$ and $M \rightarrow^*_{\beta\mu} N$, then $\boldsymbol{A}_w \sqsubseteq N$.

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

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

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

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.

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

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

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

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

We can show that these definitions coincide:

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

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

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

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

The other cases are similar.

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

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

$\boldsymbol{A}_w = x\boldsymbol{A}_w^1 \cdots \boldsymbol{A}_w^n$: Since $x\boldsymbol{A}_w^1 \cdots \boldsymbol{A}_w^n \sqsubseteq N$, $N = xN_1\cdots N_n$ for some $N_1, \ldots, N_n$, and $\boldsymbol{A}_w^i \sqsubseteq N_i$, for every $i \in \underline{n}$. Then by Definition 11.3, $\boldsymbol{A}_w^i \in \mathcal{A}^w(N_i)$, for every $i \in \underline{n}$, and by induction, $\boldsymbol{A}_w^i \in \mathcal{A}_w(N_i)$. By Lemma 9.5, there exist $M_1,\ldots,M_n$ such that $M \rightarrow_{w\mathrm{H}}^{*} xM_1\cdots M_n \rightarrow_{\beta\mu}^{*} xN_1\cdots N_n$; so in particular $M_i \rightarrow_{\beta\mu}^{*} N_i$, for every $i \in \underline{n}$. Then by Lemma 11.2, $\boldsymbol{A}_w^i \in \mathcal{A}_w(M_i)$ and by Definition 11.3, $\boldsymbol{A}_w \in \mathcal{A}^w(M)$.

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

The reverse implication of this result does not hold, since terms without whnf (which have only $\bot$ 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 $\mathcal{A}_w(M) = \{\bot\} = \mathcal{A}_w(N)$, then by Lemma 11.2 both M and N have no whnf, so $M \sim_{w\beta\mu} N$. Otherwise, either:

$x\boldsymbol{A}_w^1 \cdots \boldsymbol{A}_w^n \in \mathcal{A}_w(M) \wedge x\boldsymbol{A}_w^1 \cdots \boldsymbol{A}_w^n \in \mathcal{A}_w(N)$: By Definition 11.3 there exists $M_1,\ldots,M_n$ such that $M \rightarrow_{w\mathrm{H}}^{*} xM_1\cdots M_n$ and $\boldsymbol{A}_w^i \in \mathcal{A}_w(M_i)$. Likewise, there exist $N_1,\ldots,N_n$ such that $N \rightarrow_{w\mathrm{H}}^{*} xN_1\cdots N_n$ and $\boldsymbol{A}_w^i \in \mathcal{A}_w(N_i)$. 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 '$\rightarrow_{w\beta\mu}$', it is also under '$\rightarrow_{w\mathrm{H}}$', and we have $M \sim_{w\beta\mu} N$.

The other cases are similar.

only if : As the proof of Theorem 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, $\mathcal{A}_w(M) = \{\bot\} = \mathcal{A}_w(N)$.

$M \rightarrow^*_{w\mathrm{H}} xM_1 \cdots M_n$: Then also $N \rightarrow^*_{w\mathrm{H}} xN_1 \cdots N_n$, 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 Definition 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 $M \sim_{w\mathrm{H}} N$.

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

The other cases are similar. □

Taking '$\sqcup$' as the (partial, compatible) operation of join on terms in $\mathcal{A}_w$ generated by $\bot \sqcup \boldsymbol{A}_w = \boldsymbol{A}_w$, we can also define $\lceil M \rfloor_{\mathcal{A}_w} = \sqcup\{\, \boldsymbol{A}_w \mid \boldsymbol{A}_w \in \mathcal{A}_w(M) \,\}$; then $\lceil \cdot \rfloor_{\mathcal{A}_w}$ corresponds to the ($\lambda\mu$-variant of) Lévy-Longo trees, and it becomes easy to show that: $\lceil M \rfloor_{\mathcal{A}_w} = \lceil N \rfloor_{\mathcal{A}_w} \iff M \sim_{\mathcal{A}_w} N$. 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 $M \sim_{w\mathrm{XH}} N \iff M \sim_{w\mathrm{H}} N \iff M \sim_{\mathcal{A}_w} N \iff M \sim_{w\beta\mu} N$.

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 '$\sim_{w\mathrm{XH}}$') *For $M, N \in \lambda\mu\mathbf{x}$: $\lceil M \rfloor^{\mathrm{L}} a \approx \lceil N \rfloor^{\mathrm{L}} a$ if and only if $M \sim_{w\mathrm{XH}} N$.*

Proof : *only if* : $\lceil M \rfloor^{\mathrm{L}} a \approx \lceil N \rfloor^{\mathrm{L}} a \Rightarrow M \sim_{w\mathrm{XH}} N$.

By induction on the structure of $\lambda\mu\mathbf{x}$ terms; we distinguish the following cases.

- $\lceil M \rfloor^{\mathrm{L}} a$ can never input nor output; then $\lceil M \rfloor^{\mathrm{L}} a \approx \mathit{0} \approx \lceil N \rfloor^{\mathrm{L}} a$. Assume M has a $\rightarrow_{w\mathrm{XH}}$-normal form, then by Lemma 9.8, $\lceil M \rfloor^{\mathrm{L}} a$ is not weakly bisimilar to $\mathit{0}$; therefore, M and N both have no $\rightarrow_{w\mathrm{XH}}$-normal form, so $M \sim_{w\mathrm{XH}} N$.
- $\lceil M \rfloor^{\mathrm{L}} a \Downarrow \overline{c}$, then by Lemma 9.9, we have
  - $\lceil M \rfloor^{\mathrm{L}} a \;\approx\; \lceil \lambda x.M' \rfloor^{\mathrm{L}} c \;\triangleq\; (\nu xb)\,(\lceil M' \rfloor^{\mathrm{L}} b \mid \overline{c}\langle x,b \rangle)$,
  - $M \rightarrow^*_{w\mathrm{XH}} \lambda x.M'$ if $a = c$, or
  - $M \rightarrow^{nf}_{w\mathrm{XH}} \mu\alpha.[c]\,\lambda x.M'$ if $a \neq c$.

  Since $\lceil M \rfloor^{\mathrm{L}} a \approx \lceil N \rfloor^{\mathrm{L}} a$, also $\lceil N \rfloor^{\mathrm{L}} a \Downarrow \overline{c}$, and we have
  - $\lceil N \rfloor^{\mathrm{L}} a \;\approx\; \lceil \lambda x.N' \rfloor^{\mathrm{L}} c \;\triangleq\; (\nu xb)\,(\lceil N' \rfloor^{\mathrm{L}} b \mid \overline{c}\langle x,b \rangle)$,
  - $N \rightarrow^*_{w\mathrm{XH}} \lambda x.N'$ if $a = c$, or

– $N \rightarrow^{nf}_{w\textsc{xh}} \mu\alpha.[c]\,\lambda x.N'$ if $a \neq c$.

So we have

$$\lceil \lambda x.M' \rfloor^{\mathsf{L}} c \;\approx\; \lceil M \rfloor^{\mathsf{L}} a \;\approx\; \lceil N \rfloor^{\mathsf{L}} a \;\approx\; \lceil \lambda x.N' \rfloor^{\mathsf{L}} c$$

so in particular,

$$\lceil \lambda x.M' \rfloor^{\mathsf{L}} c \;\triangleq\; (\nu xb)\,(\lceil M' \rfloor^{\mathsf{L}} b \mid \overline{c}\langle x,b\rangle) \approx (\nu xb)\,(\lceil N' \rfloor^{\mathsf{L}} b \mid \overline{c}\langle x,b\rangle) \;\triangleq\; \lceil \lambda x.M' \rfloor^{\mathsf{L}} c;$$

then also $\lceil M' \rfloor^{\mathsf{L}} b \approx \lceil N' \rfloor^{\mathsf{L}} b$, and by induction, $M' \sim_{w\textsc{xh}} N'$; but then by the observations made above also $M \sim_{w\textsc{xh}} N$.

- If $\lceil M \rfloor^{\mathsf{L}} a \not\Downarrow_o$, but $\lceil M \rfloor^{\mathsf{L}} a \Downarrow x$, then by Lemma 9.9, $\lceil M \rfloor^{\mathsf{L}} a \approx \lceil x M_1 S_1 \cdots M_n S_n \rfloor^{\mathsf{L}} a'$ as well as $M \rightarrow^{*}_{w\textsc{xh}} x M_1 S_1 \cdots M_n S_n$. Let $S = S_1 \cdots S_n$, then we have

$$\lceil x M_1 S_1 \cdots M_n S_n \rfloor^{\mathsf{L}} a' \;\equiv (6.5)\;\; (\nu \overrightarrow{cy\alpha}\,)\,(x(u).\,!\,u{\rightarrow}\overline{c_1} \mid \overrightarrow{\lceil c_i := M_i{\cdot}c_{i+1} \rfloor^{\mathsf{L}}} \mid \lceil \boldsymbol{S} \rfloor^{\mathsf{L}})$$

where $c_n = a'$ and
$$\begin{array}{rcl}
\lceil c_i := M_i{\cdot}c_{i+1} \rfloor^{\mathsf{L}} & \triangleq & !\,c_i(v,d).(!\,\overline{v}(w).\lceil M_i \rfloor^{\mathsf{L}} w \mid !\,d{\rightarrow}\overline{c_{i+1}}) \\
\lceil \boldsymbol{S} \rfloor^{\mathsf{L}} & \equiv & \overrightarrow{\lceil y := P \rfloor^{\mathsf{L}}} \mid \overrightarrow{\lceil \alpha := Q{\cdot}\beta \rfloor^{\mathsf{L}}} \\
\lceil y_j := P_j \rfloor^{\mathsf{L}} & \triangleq & !\,\overline{y_j}(w).\lceil P_j \rfloor^{\mathsf{L}} w \\
\lceil \alpha_k := Q_k{\cdot}\beta_k \rfloor^{\mathsf{L}} & \triangleq & !\,\alpha_k(v,d).(!\,\overline{v}(w).\lceil Q_k \rfloor^{\mathsf{L}} w \mid !\,d{\rightarrow}\overline{\beta_k})
\end{array}$$

Since $\lceil M \rfloor^{\mathsf{L}} a \approx \lceil N \rfloor^{\mathsf{L}} a$, again by Lemma 9.9, $\lceil N \rfloor^{\mathsf{L}} a \approx \lceil x N_1 \cdots N_n S' \rfloor^{\mathsf{L}} a''$ and $N \rightarrow^{*}_{w\textsc{xh}} x N_1 S'_1 \cdots N_n S'_n$, with $S' = S'_1 \cdots S'_n$. Notice that

$$\lceil x N_1 S'_1 \cdots N_n S'_n \rfloor^{\mathsf{L}} a'' \;\equiv (6.5)\;\; (\nu \overrightarrow{ey\alpha}\,)\,(x(u).\,!\,u{\rightarrow}\overline{e_1} \mid \overrightarrow{\lceil e_i := N_i{\cdot}e_{i+1} \rfloor^{\mathsf{L}}} \mid \lceil S' \rfloor^{\mathsf{L}})$$

where $e_n = a''$ and
$$\begin{array}{rcl}
\lceil e_i := N_i{\cdot}e_{i+1} \rfloor^{\mathsf{L}} & \triangleq & !\,e_i(v,d).(!\,\overline{v}(w).\lceil N_i \rfloor^{\mathsf{L}} w \mid !\,d{\rightarrow}\overline{e_{i+1}}) \\
\lceil S' \rfloor^{\mathsf{L}} & \equiv & \overrightarrow{\lceil y := P' \rfloor^{\mathsf{L}}} \mid \overrightarrow{\lceil \alpha := Q'{\cdot}\beta \rfloor^{\mathsf{L}}} \\
\lceil y_j := P'_j \rfloor^{\mathsf{L}} & \triangleq & !\,\overline{y_j}(w).\lceil P'_j \rfloor^{\mathsf{L}} w \\
\lceil \alpha_k := Q'_k{\cdot}\beta_k \rfloor^{\mathsf{L}} & \triangleq & !\,\alpha_k(v,d).(!\,\overline{v}(w).\lceil Q'_k \rfloor^{\mathsf{L}} w \mid !\,d{\rightarrow}\overline{\beta_k})
\end{array}$$

Since we have

$$\lceil x M_1 S_1 \cdots M_n S_n \rfloor^{\mathsf{L}} a' \;\approx\; \lceil x N_1 S'_1 \cdots N_n S'_n \rfloor^{\mathsf{L}} a'',$$

we infer that $a' = a''$, $c_i = e_i$, and $\lceil M_i S \rfloor^{\mathsf{L}} w \approx \lceil N_i S' \rfloor^{\mathsf{L}} w$ for all $i \in \underline{n}$; then by induction, $M_i S \sim_{w\textsc{xh}} N_i S'$ for all $i \in \underline{n}$, and by the observations made above also also $M \sim_{w\textsc{xh}} N$.

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

if: $M \sim_{w\textsc{xh}} N \Rightarrow \lceil M \rfloor^{\mathsf{L}} a \approx \lceil N \rfloor^{\mathsf{L}} a$.

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

- M and N have both no $\rightarrow_{w\textsc{xh}}$-normal form, so, by Lemma 9.7, their interpretations are both weakly bisimilar to the process *0*, so in particular $\lceil M \rfloor^{\mathsf{L}} a \approx \lceil N \rfloor^{\mathsf{L}} a$; or
- both $M \rightarrow^{nf}_{w\textsc{xh}} M'$ and $N \rightarrow^{nf}_{w\textsc{xh}} N'$, and either:
  – $M' = x M_1 S_1 \cdots M_n S_n$ $(n \geq 0)$, and $N = x N_1 S'_1 \cdots N_n S'_n$, and $M_i S \sim_{w\textsc{xh}} N_i S'$, for all $i \in \underline{n}$, where $S = S_1 \cdots S_n$, $S' = S'_1 \cdots S'_n$. By Theorem 9.6, we know that both $\lceil M \rfloor^{\mathsf{L}} a \approx \lceil x M_1 S_1 \cdots M_n S_n \rfloor^{\mathsf{L}} a$ and $\lceil N \rfloor^{\mathsf{L}} a \approx \lceil x N_1 S'_1 \cdots N_n S'_n \rfloor^{\mathsf{L}} a$. Notice that

$$\lceil x M_1 S_1 \cdots M_n S_n \rfloor^{\mathsf{L}} a \;\equiv (6.5)\;\; (\nu \overrightarrow{y\alpha c}\,)\,(\lceil x \rfloor^{\mathsf{L}} c_1 \mid \overrightarrow{\lceil c_i := M_i{\cdot}c_{i+1} \rfloor^{\mathsf{L}}} \mid \lceil \boldsymbol{S} \rfloor^{\mathsf{L}})$$

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

and similarly for $\lceil x N_1 S'_1 \cdots N_n S'_n \rfloor^{\mathsf{L}} a$. By induction (see Definition 10.4),

$$(\nu \overline{y}\overrightarrow{\alpha}\,)\,(\ulcorner M_i \lrcorner^{\mathsf{L}} w \mid \ulcorner \boldsymbol{S} \lrcorner^{\mathsf{L}}) \;\triangleq\; \ulcorner M_i S \lrcorner^{\mathsf{L}} w \;\approx\; \ulcorner N_i S' \lrcorner^{\mathsf{L}} w \;\triangleq\; (\nu \overline{y}\overrightarrow{\alpha}\,)\,(\ulcorner N_i \lrcorner^{\mathsf{L}} w \mid \ulcorner S' \lrcorner^{\mathsf{L}})$$

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

$$\begin{array}{r}(\nu \overline{y}\overrightarrow{\alpha}\,)\,(!\,c_i(v,d).(!\,\overline{v}(w).\ulcorner M_i \lrcorner^{\mathsf{L}} w \mid !\,d \rightarrow \overline{c_{i+1}}) \mid \ulcorner \boldsymbol{S} \lrcorner^{\mathsf{L}}) \;\approx \qquad\qquad\qquad\qquad \\ (\nu \overline{y}\overrightarrow{\alpha}\,)\,(!\,c_i(v,d).(!\,\overline{v}(w).\ulcorner N_i \lrcorner^{\mathsf{L}} w \mid !\,d \rightarrow \overline{c_{i+1}}) \mid \ulcorner S' \lrcorner^{\mathsf{L}})\end{array}$$

so also $\ulcorner x M_1 S_1 \cdots M_n S_n \lrcorner^{\mathsf{L}} a \approx \ulcorner x N_1 S'_1 \cdots N_n S'_n \lrcorner^{\mathsf{L}} a$ but then also $\ulcorner M \lrcorner^{\mathsf{L}} a \approx \ulcorner N \lrcorner^{\mathsf{L}} a$.

- $M' = \lambda x.M''$, $N' = \lambda x.N''$, and $M'' \sim_{w\mathrm{XH}} N''$. By Theorem 9.6, we have $\ulcorner M \lrcorner^{\mathsf{L}} a \approx \ulcorner \lambda x.M'' \lrcorner^{\mathsf{L}} a$ and $\ulcorner N \lrcorner^{\mathsf{L}} a \approx \ulcorner \lambda x.N'' \lrcorner^{\mathsf{L}} a$. Notice that

$$\begin{array}{rcll}\ulcorner \lambda x.M'' \lrcorner^{\mathsf{L}} a & \triangleq & (\nu x b)\,(\ulcorner M'' \lrcorner^{\mathsf{L}} b \mid \overline{a}\langle x,b\rangle) & \text{and} \\ \ulcorner \lambda x.N'' \lrcorner^{\mathsf{L}} a & \triangleq & (\nu x b)\,(\ulcorner N'' \lrcorner^{\mathsf{L}} b \mid \overline{a}\langle x,b\rangle) & \end{array}$$

  By induction, $\ulcorner M'' \lrcorner^{\mathsf{L}} b \approx \ulcorner N'' \lrcorner^{\mathsf{L}} b$. As above, since '≈' is a congruence, also $\ulcorner M \lrcorner^{\mathsf{L}} a \approx \ulcorner N \lrcorner^{\mathsf{L}} a$.

- $M' = \mu\gamma.[\delta]M''$, $N' = \mu\gamma.[\delta]N''$. Then M'' and N'' themselves are in normal form and $M'' \sim_{w\mathrm{XH}} N''$. By Theorem 9.6, $\ulcorner M \lrcorner^{\mathsf{L}} a \approx \ulcorner \mu\gamma.[\delta]M'' \lrcorner^{\mathsf{L}} a$ and $\ulcorner N \lrcorner^{\mathsf{L}} a \approx \ulcorner \mu\gamma.[\delta]N'' \lrcorner^{\mathsf{L}} a$. Notice that

$$\begin{array}{rcccll}\ulcorner \mu\gamma.[\delta].M'' \lrcorner^{\mathsf{L}} a & \triangleq & \ulcorner M'' \lrcorner^{\mathsf{L}} \delta\,\{a/\gamma\} & \triangleq & \ulcorner M''\,\{\alpha/\gamma\} \lrcorner^{\mathsf{L}} \delta & \text{and} \\ \ulcorner \mu\gamma.[\delta].N'' \lrcorner^{\mathsf{L}} a & \triangleq & \ulcorner N'' \lrcorner^{\mathsf{L}} \delta\,\{a/\gamma\} & \triangleq & \ulcorner N''\,\{\alpha/\gamma\} \lrcorner^{\mathsf{L}} \delta & \end{array}$$

  By induction, $\ulcorner M''\,\{\alpha/\gamma\} \lrcorner^{\mathsf{L}} \delta \approx \ulcorner N''\,\{\alpha/\gamma\} \lrcorner^{\mathsf{L}} \delta$; since '≈' is a congruence, $\ulcorner M \lrcorner^{\mathsf{L}} a \approx \ulcorner N \lrcorner^{\mathsf{L}} a$. □

We can now prove our main result:

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

Proof: By Corollary 11.8 and Theorem 12.1. □

13 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 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 Theorem 7.1 (only) to show that the encoding models the reduction steps (where N gets distributed):

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

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 13.1 (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 '$\rightarrow_{\mathrm{X}}$' on terms in $\lambda\mathbf{x}$ is defined by the rules:

$$
\begin{array}{rcl}
(\lambda x.M)\,N & \to & M\langle x := N\rangle \\
(\lambda y.M)\,\langle x := L\rangle & \to & \lambda y.(M\langle x := L\rangle) \\
(MN)\,\langle x := L\rangle & \to & (M\langle x := L\rangle)\,(N\langle x := L\rangle) \\
x\,\langle x := L\rangle & \to & L \\
M\langle x := L\rangle & \to & M \quad (x \not\in fv(M))
\end{array}
\qquad
M \to N \;\Rightarrow\; \left\{
\begin{array}{lcl}
ML & \to & NL \\
LM & \to & LN \\
\lambda x.M & \to & \lambda x.N \\
M\langle x := L\rangle & \to & N\langle x := L\rangle \\
L\langle x := M\rangle & \to & L\langle x := N\rangle
\end{array}\right.
$$

Definition 13.2 (Explicit head and lazy reduction) *i*) *Explicit head-reduction* '$\to_{\mathrm{XH}}$' on $\lambda\mathbf{x}$ is defined by:

$$
\begin{array}{rrcll}
(\beta): & (\lambda x.M)\,N & \to & M\langle x := N\rangle & \\
(hv): & xS_0M_1S_1\cdots M_nS_{n+1} & \to & NS_0M_1S_1\cdots M_nS_{n+1} & (n \geq 0,\ \langle x := N\rangle \in S_{n+1}) \\
(\lambda S): & (\lambda y.M)S & \to & \lambda y.(MS) & \\
(gc): & MS & \to & MS\backslash x & (x \in S, x \not\in M)
\end{array}
$$

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

ii) We define *explicit lazy reduction* '$\to_{\mathrm{XL}}$' by eliminating, from '$\to_{\mathrm{XH}}$', the rules

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

Notice that lazy reduction does not correspond to weak explicit head reduction, since lazy reduction does not allow substitutions to be propagated under abstractions.

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

Definition 13.3 We extend Milner's interpretation (see Definition 3.1) to $\lambda\mathbf{x}$ by adding the case:

$$
\ulcorner M\langle x := N\rangle \lrcorner^{\mathrm{M}}\, a \;\stackrel{\Delta}{=}\; (\nu x)\,(\ulcorner M \lrcorner^{\mathrm{M}}\, a \mid \ulcorner x := N \lrcorner^{\mathrm{M}})
$$

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

Theorem 13.4 ($\ulcorner\cdot\lrcorner^{\mathrm{M}}$ preserves $\to_{\mathrm{XL}}$) *If $M \to_{\mathrm{XL}}^{*} N$, then $\ulcorner M \lrcorner^{\mathrm{M}}\, a \to_{\pi}^{*}, \approx_{\mathrm{G}} \ulcorner N \lrcorner^{\mathrm{M}}\, a$.*

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

$(\lambda x.M)N \to M\langle x := N\rangle$: $\ulcorner(\lambda x.M)\,N \lrcorner^{\mathrm{M}}\, a \;\to_{\pi}^{+}$ (3.2) $(\nu x)\,(\ulcorner M \lrcorner^{\mathrm{M}}\, a \mid \ulcorner x := N \lrcorner^{\mathrm{M}}) \;\stackrel{\Delta}{=}$
$\ulcorner M\langle x := N\rangle \lrcorner^{\mathrm{M}}\, a$

$xS_0M_1S_1\cdots M_nS_n \to NS_0M_1S_1\cdots M_nS_n$:

$$
\begin{array}{lll}
\ulcorner xS_0M_1S_1\cdots M_nS_n \lrcorner^{\mathrm{M}}\, a & \stackrel{\Delta}{=} & \\
(\nu c_n \vec{y}_n)\,(\ulcorner xS_0M_1S_1\cdots M_{n-1}S_{n-1} \lrcorner^{\mathrm{M}}\, c_n \mid (\nu z)\,(\overline{c_n}\, z.\overline{c_n}\, a.\ulcorner z := M_n \lrcorner^{\mathrm{M}}) \mid \ulcorner S_n \lrcorner^{\mathrm{M}}) & \stackrel{\Delta}{=} & \\
(\nu \vec{c_i}\; \vec{\vec{y}}_i)\,(\ulcorner x \lrcorner^{\mathrm{M}}\, c_1 \mid \overrightarrow{(\nu z)\,(\overline{c_i}\, z.\overline{c_i}\, c_{i+1}.\ulcorner z := M_i \lrcorner^{\mathrm{M}})} \mid \overrightarrow{\ulcorner S_i \lrcorner^{\mathrm{M}}}) & \equiv, \stackrel{\Delta}{=} & (c_{n+1} = a) \\
(\nu \vec{c_i}\; \vec{\vec{y}}_i)\,(\overline{x}\, c_1 \mid \overrightarrow{(\nu z)\,(\overline{c_i}\, z.\overline{c_i}\, c_{i+1}.\ulcorner z := M_i \lrcorner^{\mathrm{M}})} \mid x(w).\ulcorner N \lrcorner^{\mathrm{M}}\, w \mid \overrightarrow{\ulcorner S_i \lrcorner^{\mathrm{M}}}) & \to_{\pi}(x) & \\
(\nu \vec{c_i}\; \vec{\vec{y}}_i)\,(\ulcorner N \lrcorner^{\mathrm{M}}\, c_1 \mid \overrightarrow{(\nu z)\,(\overline{c_i}\, z.\overline{c_i}\, c_{i+1}.\ulcorner z := M_i \lrcorner^{\mathrm{M}})} \mid \overrightarrow{\ulcorner S_i \lrcorner^{\mathrm{M}}}) & \stackrel{\Delta}{=} & \\
\ulcorner NS_0M_1S_1\cdots M_nS_n \lrcorner^{\mathrm{M}}\, a & &
\end{array}
$$

$MS \to MS\backslash x,\ x \in S, x \not\in M$: $\ulcorner MS \lrcorner^{\mathrm{M}}\, a \;\stackrel{\Delta}{=}\; (\nu \vec{y}\,)\,(\ulcorner M \lrcorner^{\mathrm{M}}\, a \mid \ulcorner S \lrcorner^{\mathrm{M}}) \;=\; (\nu \vec{y}\,)\,(\ulcorner M \lrcorner^{\mathrm{M}}\, a \mid \ulcorner S\backslash x \lrcorner^{\mathrm{M}} \mid \ulcorner S_x \lrcorner^{\mathrm{M}}) \;\equiv$
$(\nu \vec{y}\,\backslash x)\,(\ulcorner M \lrcorner^{\mathrm{M}}\, a \mid \ulcorner S\backslash x \lrcorner^{\mathrm{M}}) \mid (\nu x)\,(\ulcorner S_x \lrcorner^{\mathrm{M}}) \;\approx_{\mathrm{G}}\; (\nu \vec{y}\,\backslash x)\,(\ulcorner M \lrcorner^{\mathrm{M}}\, a \mid \ulcorner S\backslash x \lrcorner^{\mathrm{M}}) \;\stackrel{\Delta}{=}\; \ulcorner MS\backslash x \lrcorner^{\mathrm{M}}\, a$ □

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

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

Corollary 13.5 If M is closed, and $M \rightarrow^{nf}_{\mathrm{XL}} (\lambda y.N)\,\langle \overline{x:=\,}\,\rangle N$, then $\ulcorner M \lrcorner^{\mathrm{M}} a \rightarrow^{nf}_{\pi}, \approx_{\mathrm{G}} (\nu \vec{x}\,)\,(\ulcorner \lambda x.N \lrcorner^{\mathrm{M}} a \mid \overrightarrow{\ulcorner x := N \lrcorner^{\mathrm{M}}}\,)$.

The restriction of our encoding $\ulcorner\cdot\lrcorner^{\mathrm{L}}\cdot$ from Definition 6.1 to $\lambda\mathbf{x}$ is defined by:

Definition 13.6 (Output-based encoding of $\lambda\mathbf{x}$-terms in π) The mapping $\ulcorner\cdot\lrcorner^{\lambda}\cdot$ is defined by:

$$\begin{array}{rcll}
\ulcorner x \lrcorner^{\lambda} a & \stackrel{\Delta}{=} & x(u).u \rightarrow \overline{a} & (x \neq a) \\
\ulcorner \lambda x.M \lrcorner^{\lambda} a & \stackrel{\Delta}{=} & (\nu xb)\,(\ulcorner M \lrcorner^{\lambda} b \mid \overline{a}\langle x,b\rangle) & (a,b\ \mathit{fresh}) \\
\ulcorner MN \lrcorner^{\lambda} a & \stackrel{\Delta}{=} & (\nu c)\,(\ulcorner M \lrcorner^{\lambda} c \mid c(v,d).(\ulcorner v := N \lrcorner^{\lambda} \mid d \rightarrow \overline{a})) & (a,b,c,v,d\ \mathit{fresh}) \\
\ulcorner M\,\langle x := N\rangle \lrcorner^{\lambda} a & \stackrel{\Delta}{=} & (\nu x)\,(\ulcorner M \lrcorner^{\lambda} a \mid \ulcorner x := N \lrcorner^{\lambda}\,) & (a\ \mathit{fresh}) \\
\ulcorner x := N \lrcorner^{\lambda} & \stackrel{\Delta}{=} & !\,\overline{x}(w).\ulcorner N \lrcorner^{\lambda} w & (w\ \mathit{fresh})
\end{array}$$

Notice the absence of replication, compared to the definition of $\ulcorner\cdot\lrcorner^{\mathrm{L}}\cdot$, 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 Theorem 12.1, when considering $\ulcorner M \lrcorner^{\lambda} a \Downarrow \overline{c}$, 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 the distributive character of implicit substitution.

Notice also the difference in the interpretation of explicit substitution with the one defined in [12], which uses

$$\ulcorner x := N \lrcorner^{\mathrm{s}} a \;\stackrel{\Delta}{=}\; !\,\ulcorner N \lrcorner^{\mathrm{s}} x$$

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 Theorem 8.4, also for our encoding we can show:

Theorem 13.7 *If M is a closed λ-term, and $M \rightarrow^{nf}_{w\mathrm{XH}} N$ then $\ulcorner M \lrcorner^{\lambda} a \rightarrow^{nf}_{\pi}, \approx_{\mathrm{G}} \ulcorner N \lrcorner^{\lambda} a$.*

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, Theorem 3.3, but for the fact that our result is stated with head reduction.

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

Theorem 13.8 *i) If $M \in \lambda\mathbf{x}$, and $M \rightarrow_{\mathrm{XH}} N$, then $\ulcorner M \lrcorner^{\lambda} a \rightarrow^{*}_{\pi}, \approx_{\mathrm{R}}, \approx_{\mathrm{G}} \ulcorner N \lrcorner^{\lambda} a$.*

ii) If $M \in \lambda\mathbf{x}$, and $\ulcorner M \lrcorner^{\lambda} a \rightarrow_{\pi} \mathsf{P}$, then there exists N such that $\mathsf{P} \approx_{\mathrm{RG}} \ulcorner N \lrcorner^{\lambda} a$, and $M \rightarrow_{\mathrm{XH}} N$.

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

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

v) If $M \in \lambda\mathbf{x}$, and $M \rightarrow_{\mathrm{XH}}^{nf} N$, then $\ulcorner M \lrcorner^{\lambda} a \Downarrow_{\pi}$.

vi) If $M \in \lambda$, and $M \rightarrow_{\beta}^{hnf} N$, then $\ulcorner M \lrcorner^{\lambda} a \Downarrow_{\pi}$.

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

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 using the approach we have used above, we can show:

Theorem 13.9 *For $M,N \in \lambda$, $\ulcorner M \lrcorner^{\lambda} a \approx \ulcorner N \lrcorner^{\lambda} a \iff M \sim_{w\mathrm{XH}} N \iff M \sim_{w\mathrm{H}} N \iff M \sim_{\mathcal{A}_w} N \iff M \sim_{w\beta} N$.*

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

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 '$\rightarrow_{\mathrm{XH}}$' 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 '$\rightarrow_{\mathrm{XH}}$'. 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, '$\rightarrow_{w\mathrm{XH}}$'. 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 '$\sim_{w\beta\mu}$', '$\sim_{w\mathrm{H}}$', '$\sim_{w\mathrm{XH}}$', and '$\sim_{\mathcal{A}_w}$' on $\lambda\mu$ terms, and have shown that these all coincide on *pure* terms (without explicit substitution). We have proved that $M \sim_{w\mathrm{XH}} N \iff \ulcorner M \lrcorner^{\mathrm{L}} a \approx \ulcorner N \lrcorner^{\mathrm{L}} a$, 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 improved our paper, and 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 $\overline{\lambda}\mu\tilde{\mu}$-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] 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.

[46] 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.

[47] 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.

[48] 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.

[49] W. Py. *Confluence en $\lambda\mu$-calcul*. Thèse de doctorat, Université de Savoie, 1998.

[50] D. Sangiorgi. The Lazy Lambda Calculus in a Concurrency Scenario. *Information and Computation*, 111(1):120–153, 1994.

[51] D. Sangiorgi and D. Walker. *The Pi-Calculus*. Cambridge University Press, 2001.

[52] 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.

[53] 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.

[54] Th. Streicher and B. Reus. Classical logic: Continuation Semantics and Abstract Machines. *Journal of Functional Programming*, 11(6):543–572, 1998.

[55] H. Thielecke. *Categorical Structure of Continuation Passing Style*. PhD thesis, University of Edinburgh, 1997. LFCS technical report ECS-LFCS-97-376.

[56] 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.

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