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

Steffen van Bakel[1] and Maria Grazia Vigliotti[2]

[1]: Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2BZ, UK
[2]: Rail Safety and Standards Board, Block 2, Angel Square, 1 Torrens Street, London EC1V 1NY
s.vanbakel@imperial.ac.uk mgv@rbbs.com

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. Using a notion of implicative type-context assignment for the π-calculus, we also show that assignable types are preserved by the interpretation. 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.

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 [21, 14], Parigot's $\lambda\mu$-calculus [39], or $\Lambda\mu$[1] [26], 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 [37].

$\lambda\mu$ 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. 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 the 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 [27] and [9], where the interpretation of $\lambda\mu$ into $\lambda\mu\tilde{\mu}$ and $\mathcal{X}$, respectively, does not respect reduction.

[1] The name $\Lambda\mu$ was first introduced in [45], 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.

Over the last two decades, the π-calculus and its dialects have proven to give an interesting and expressive model of computation. Encodings of variants of the pure λ-calculus [21, 14] started with Milner's encoding of the lazy λ-calculus [37] and quickly led to more thorough investigations (see [18, 44, 10]; many more papers were written on the topic), also in the direction of object oriented calculi [29, 44]. The strength of the results that have been shown in those papers – like soundness, completeness, termination, and full abstraction – has encouraged researchers to investigate interpretations into the π-calculus of various calculi that have their foundation in classical logic, as done in, for example, [30, 7, 22, 15]. From these papers it might seem that the interpretation of such 'classical' calculi comes at a great expense; for example, to encode *typed* $\lambda\mu$, [30] defines an extension of Milner's encoding and considers a version of the π-calculus that is strongly typed; [7] shows preservation of reduction in $\mathcal{X}$ only with respect to the contextual ordering $\sqsubseteq_{\mathsf{c}}$ (so not with respect to contextual equivalence $\sim_{\mathrm{C}}$, nor with respect to weak bisimilarity $\approx$); [22] defines a non-compositional interpretation of $\lambda\mu\tilde{\mu}$ that strongly depends on recursion, and does not regard the logical aspect at all.

Here 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 [10] and is a natural variant of that for $\Lambda\mu$ we presented in [11]; our approach was compared to the traditional input-based one in [28]. In those papers, we have shown that those encodings respects *single-step explicit head-reduction* $\rightarrow_{\mathrm{xH}}$ (which only ever replaces the head variable of a term, see Definition 5.2) modulo contextual equivalence $\sim_{\mathrm{C}}$; here we restate those properties with respect to weak bisimilarity $\approx$. We show that by extending the output-based interpretation $\llbracket M \rrbracket\, a$ of λ-terms [10] (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 perfectly possible to encode $\Lambda\mu$ and our first results in this direction were indeed on that calculus [11]; as we will argue below, to achieve full abstraction here we have to focus on $\lambda\mu$.

To accurately define the notion of reduction that is modelled by our interpretation, following [10], in [11] we defined (untyped) $\Lambda\mu\mathbf{x}$, a version with explicit substitution [1, 17] 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 [37, 44] has been strongly argued by us in [10, 11], and makes an important contribution here as well. In [10] 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.

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 guards, thereby the calculus that can be effectively represented is limited (note that this restriction is dropped in [30]); also other interpretations defined in the past do not model full reduction for the same reason. In our case, as in [10], 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 [10] (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 substitu-

tion is dealt with entirely by congruence in π (see also Example 6.7), 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 6.3).

We will show a number of results we also showed in [10, 11] for the λ and $\Lambda\mu$ calculi, respectively. Using a notion of functional type assignment for the π-calculus, we show in Theorem 6.10 that the encoding respects assignable types, using the notion of functional type assignment for the π-calculus we defined in [7]. In Theorem 7.1, we show that single-step explicit head reduction is respected by the encoding in such a way that each β-reduction step is implemented through at least one synchronisation; this leads to an operational soundness and completeness result. In Theorem 7.5 we show that the encoding also respects equality on $\lambda\mathbf{x}$, but modulo weak bisimulation, and in Theorem 7.6 that it gives a semantics for $\lambda\mu$.

As for full abstraction (*i.e.* two terms that are equivalent under the interpretation are also operational equivalent), Sangiorgi has shown a full abstraction result [43, 44] for (essentially) Milner's encoding $\llbracket M \rrbracket^{\mathrm{M}}_{\mathbb{I}}\, a$, by showing that $\llbracket M \rrbracket^{\mathrm{M}}_{\mathbb{I}}\, a \approx \llbracket N \rrbracket^{\mathrm{M}}_{\mathbb{I}}\, a$ if and only if $M \cong N$, where $\cong$ 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 we will use a new, considerably different technique. First we characterise what is exactly the equivalence between terms in $\lambda\mu$ that is representable in the π-calculus through $\llbracket \cdot \rrbracket \cdot$; this turns out to be *weak* equivalence (see Section 9), that essentially equates terms that have the same $\lambda\mu$-Lévy-Longo tree [35, 36] (a lazy variant of Böhm trees [14]).[2] We will show that our interpretation respects $\rightarrow_{\mathbf{x}\mathrm{H}}$ modulo $\approx$ and extend this result to *weak explicit head equivalence* $\sim_{w\mathbf{x}\mathrm{H}}$, the equivalence relation generated by $\rightarrow_{\mathbf{x}\mathrm{H}}$ that equates also terms without weak head-normal form. The main proof of the full abstraction result is then achieved through showing that $\sim_{w\mathbf{x}\mathrm{H}}$ 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 the latter result, we define a choice of operational equivalences for the $\lambda\mu$-calculus, both with and without explicit substitution. Next to $\sim_{w\mathbf{x}\mathrm{H}}$ we define *weak head equivalence* $\sim_{w\mathrm{H}}$ and show that for $\lambda\mu$-terms without explicit substitution, $\sim_{w\mathbf{x}\mathrm{H}}$ corresponds to $\sim_{w\mathrm{H}}$. Following essentially [48, 49], 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.

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* 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.

Working towards our full abstraction result, in Section 8 we will define notions of weak reduction, in particular *weak head reduction* and *weak explicit head reduction*. In Section 9 we define the two notions of equivalence these induce, also equating terms without weak head-normal form and show that these notions coincide on pure $\lambda\mu$ terms (*i.e.* without explicit

[2] In that sense, Sangiorgi's full abstraction result [44] and ours do correspond.

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 10, we define a notion of *weak approximation* for $\lambda\mu$, and show the semantics this induces is fully abstract with respect to both $\sim_{w\mathrm{H}}$ and $\sim_{w\beta\mu}$.

Then, in Section 11, we show that our logical interpretation is fully abstract with respect to weak bisimilarity $\approx$ on processes and $\sim_{w\mathbf{x}\mathrm{H}}$, $\sim_{w\mathrm{H}}$, $\sim_{\mathcal{A}_w}$, and $\sim_{w\beta\mu}$ on pure $\lambda\mu$-terms.

This paper is an extended version of [11, 13], but dealing with $\lambda\mu$ (rather than $\Lambda\mu$ as in [11]).

Notation: We will write $\underline{n}$ for the set $\{1,\ldots,n\}$. We will use a vector notation $\vec{\cdot}$ as abbreviation for any sequence: for example, $\vec{x_i}$ stands for $x_1,\ldots,x_n$, for any irrelevant n, or for $\{x_1,\ldots,x_n\}$, and $\langle\overline{\alpha_i := N_i \cdot \beta_i}\rangle$ for $\langle\alpha_1 := N_1 \cdot \beta_1\rangle \cdots \langle\alpha_n := N_n \cdot \beta_n\rangle$, $\overline{M_i =_{\mathrm{H}} N_i}$ for $\forall i \in \underline{n}\,[M_i =_{\mathrm{H}} N_i]$, etc. When possible, we will drop the indices.

1 The $\lambda\mu$ calculus

In this section, we will briefly discuss Parigot's $\lambda\mu$-calculus [39]; we assume the reader to be familiar with the λ-calculus and its notion of reduction $\rightarrow_\beta$ and equality $=_\beta$, so will be brief on details. In the next section we will define explicit head-reduction for $\lambda\mu\mathbf{x}$, a variant of $\lambda\mu$ with explicit substitution *à la* $\lambda\mathbf{x}$ [17], and will show full abstraction results for $\lambda\mu\mathbf{x}$; since $\lambda\mu\mathbf{x}$ implements $\lambda\mu$-reduction, this implies that, automatically, our main results are also shown for standard reduction (with implicit substitution).

$\lambda\mu$ is a proof-term syntax for classical logic, expressed in Natural Deduction, defined as an extension of the Curry type assignment system for the λ-calculus by adding 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:

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

We will occasionally write $\mathtt{C}$ for the pseudo-term $[\alpha]M$.

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

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

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

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

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.

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

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

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

$$\begin{array}{rcll}
([\alpha]M)\,[N{\cdot}\gamma/\alpha] & \triangleq & [\gamma](M\,[N{\cdot}\gamma/\alpha])N & \\
([\beta]M)\,[N{\cdot}\gamma/\alpha] & \triangleq & [\beta](M\,[N{\cdot}\gamma/\alpha]) & (\alpha \neq \beta) \\
(\mu\beta.\mathsf{C})\,[N{\cdot}\gamma/\alpha] & \triangleq & \mu\beta.(\mathsf{C}\,[N{\cdot}\gamma/\alpha]) & \\
x\,[N{\cdot}\gamma/\alpha] & \triangleq & x & \\
(\lambda x.M)\,[N{\cdot}\gamma/\alpha] & \triangleq & \lambda x.(M\,[N{\cdot}\gamma/\alpha]) & \\
(M_1M_2)\,[N{\cdot}\gamma/\alpha] & \triangleq & M_1\,[N{\cdot}\gamma/\alpha]\ M_2\,[N{\cdot}\gamma/\alpha] &
\end{array}$$

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

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

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

as well as the *simplification rules*:

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

[3] This notion is often defined as $M[N/\alpha]$, where every (pseudo) sub-term of the form $[\alpha]M'$ is replaced by $[\alpha]M'N$; in our opinion, this creates confusion, since α gets 'reintroduced'; it is in fact a new name, which is illustrated by the fact that, in the typed version α then changes type. Moreover, when making this substitution explicit, bound and free occurrences of the same name would be introduced, violating Barendregt's convention.

which are added mainly to simplify the presentation of results. We use the contextual rules:[4]

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

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

That this notion of reduction is confluent was shown in [42]; so we have:

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

For convenience, Parigot also considers $[\alpha]M$ and $\mu\alpha.M$ as *pseudo*-terms, that represent *de-activation* and *activation*.[6]

The intuition behind the structural rule is given by [26]: *"in a $\lambda\mu$-term $\mu\alpha.M$ of type $A{\to}B$, only the subterms named by α are* really *of type $A{\to}B$ (...); hence, when such a μ-abstraction is applied to an argument, this argument must be passed over to the sub-terms named by α."* We can think of $[\alpha]M$ as storing the type of M amongst the alternative conclusions by giving it the name α.

[40] has shown that typeable terms are strongly normalisable. It also defines the extensional rules:

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

Here we do not consider these rules: the model we present through our interpretation is not extensional, and we can therefore not show that those rules are preserved by the interpretation (see Remark 6.8).

Example 1.6 As an example illustrating the fact that this system is more powerful than the system for the λ-calculus, here is a proof that it is possible to inhabit Peirce's Law (due to [38]):

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

The underlying logic of the system of Definition 1.2 corresponds to *minimal classical logic* [5].

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

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

[6] In fact, [39] formulates the renaming rule as $[\beta](\mu\gamma.M) \to M[\beta/\gamma]$.

We also consider the notion of head reduction; it is defined in [48] for the λ-calculus by first defining the head-redex of a term as the subterm $(\lambda x.M)N$ in a term of the form

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

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

$$\lambda x_1 x_2 \cdots x_n.yL_1 L_2 \cdots L_m \quad (n \geq 0, m \geq 0)$$

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

Definition 1.7 (Head reduction for $\lambda\mu$ (*cf.* [34])) *i*) We define *head reduction* $\to_{\mathrm{H}}$ as the restriction of $\to_{\beta\mu}$ by removing the contextual rule:

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

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

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

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

The following is straightforward:

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

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

2 The synchronous π-calculus with pairing

The notion of π-calculus that we consider in this paper was already considered in [10] and is different from other systems studied in the literature [29] in that it adds *pairing* and uses a *let*-construct to deal with inputs of pairs of names that get distributed, similar to that defined in [2]; in contrast to [7, 10], we do not consider the asynchronous π-calculus.

As already argued in [10], the main reason for the addition of pairing lies in preservation of (implicate, or functional) type assignment;[7] therefore *data* is introduced as a structure over names, such that not only names but also pairs of names can be sent (but not a pair of pairs); this way a channel may pass along either a name or a pair of names.

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

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

[7] If we would not consider types in this paper, we could consider the standard π-calculus; however, the details of the interpretation would change (more replication would be needed; for details see [12]).

Processes are defined by:

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

We see, as usual, ν as a binder, and call the name n *bound* in $(\nu n)\,P$, x bound in $a(x).P$ and x,y bound in $\mathit{let}\,\langle x,y\rangle = p \ \mathit{in}\ P$; we write $bn(P)$ for the set of bound names in P; n is *free* in P if it occurs in P but is not bound, and we write $fn(P)$ for the set of free names in P. We call a in $(\nu a)\,P$ a *hidden* channel. A *context* $\mathtt{C}[\cdot]$ is a process with a hole $[]$; we call $a(x)$ and $\overline{a}\,p$ *guards*, and call P in $a(x).P$ and $\overline{a}\,p.P$ a process *under guard*.

Notice that the pairing in data is *not* recursive.

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

We abbreviate $a(x).\mathit{let}\,\langle y,z\rangle = x \ \mathit{in}\ P$ by $a(y,z).P$, as well as $(\nu m)\,(\nu n)\,P$ by $(\nu mn)\,P$, and write $\overline{a}\,p$ for $\overline{a}\,p.0$. As in [44], we write $a{\rightarrow}b$ for the *forwarder* $a(x).\overline{b}\,x$ and $\overline{x}(w).P$ for $(\nu w)\,(\overline{x}\,w.P)$.

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 of rule $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$, 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 is defined by the following (elementary) rules:

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

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

ii) We say that a P is *irreducible* (is in *normal form*) if it does not contain a possible synchroni-

sation, *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,

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

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

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

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

iii) A *barbed bisimilarity* $\dot{\approx}$ is the largest symmetric relation such that $P \dot{\approx} Q$ 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 $P \rightarrow_\pi^* P'$, then there exists Q' such that $Q \rightarrow_\pi^* Q'$ and $P' \dot{\approx} Q'$;

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

The following is easy to show.[8]

Proposition 2.5 *Let P, Q not contain a and $a \neq b$, then*

$$\begin{array}{rcl} (\nu a)\,(\overline{a}\,p.P \mid a(x).Q) & \approx & P \mid Q[p/x] \\ (\nu a)\,(!\overline{a}\,p.P \mid a(x).Q) & \approx & P \mid Q[p/x] \end{array}$$

This expresses that synchronisation over hidden (internal) channels is unobservable.

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

Lemma 2.6 ([12]) *Let $x \neq c$ only be used as input channel in P and Q, and not appear in R, then:*

$$\begin{array}{rcll} (\nu x)\,(P \mid Q \mid !\overline{x}(w).R) & \approx & (\nu x)\,((\nu y)\,(P[y/x] \mid !\overline{y}(w).R) \mid Q \mid !\overline{x}(w).R) & y\ \mathit{fresh} \\ (\nu x)\,(P \mid Q \mid !\overline{x}(w).R) & \approx & (\nu x)\,(P \mid !\overline{x}(w).R) \mid (\nu x)\,(Q \mid !\overline{x}(w).R) & \\ (\nu x)\,(!c(v,d).P \mid !\overline{x}(w).R) & \approx & !c(v,d).((\nu x)\,(P \mid !\overline{x}(w).R)) & \\ (\nu x)\,(!\overline{c}(y).P \mid !\overline{x}(w).R) & \approx & !\overline{c}(y).((\nu x)\,(P \mid !\overline{x}(w).R)) & \end{array}$$

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

$$\begin{array}{rcl} (\nu x)\,(P \mid Q \mid !x(v,d).R) & \approx & (\nu x)\,(P \mid !x(v,d).R) \mid (\nu x)\,(Q \mid !x(v,d).R) \\ (\nu x)\,(!c(y).P \mid !x(v,d).R) & \approx & !c(y).((\nu x)\,(P \mid !x(v,d).R)) \\ (\nu x)\,(!\overline{c}(y).P \mid !x(v,d).R) & \approx & !\overline{c}\,y.((\nu x)\,(P \mid !x(v,d).R)) \end{array}$$

Proof: Straightforward. □

The following properties follow directly from the definition of $\approx$ and Proposition 2.5:

[8] This property was stated in [11] using contextual equivalence rather than weak bisimilarity.

Proposition 2.7 *i) If for all P', Q' such that $P \rightarrow_\pi^* P'$, $Q \rightarrow_\pi^* Q'$ over hidden channels, we have $P' \approx Q'$, then $P \approx Q$.*

ii) Let P, Q be such that no interaction is possible between them, then: $P \approx P'$ and $Q \approx Q'$ if and only if $P \mid Q \approx P' \mid Q'$.

The π-calculus is equipped with a rich type theory [44], from the basic type system for counting the arity of channels [41] to session types [31] and sophisticated linear types in [30]. The notion of type assignment we use here is the one first defined in [7] and differs from systems presented in the past in that types do not contain channel information, and in that it expresses *implication*, *i.e.* has functional types and describes the '*input-output interface*' of a process.

Definition 2.8 (Implicative IO type assignment for π [7, 8]) Type assignment for the π-calculus is defined by the following sequent system:[9]

$$(0):\ \frac{}{0 : \Gamma \vdash \Delta} \qquad (in):\ \frac{P : \Gamma, x{:}A \vdash x{:}A, \Delta}{a(x).P : \Gamma, a{:}A \vdash \Delta} \qquad (out):\ \frac{P : \Gamma, b{:}A \vdash b{:}A, \Delta}{\overline{a}\,b.P : \Gamma, b{:}A \vdash a{:}A, b{:}A, \Delta}\ (a \neq b)$$

$$(!):\ \frac{P : \Gamma \vdash \Delta}{!\,P : \Gamma \vdash \Delta} \qquad (\nu):\ \frac{P : \Gamma, a{:}A \vdash a{:}A, \Delta}{(\nu a)\,P : \Gamma \vdash \Delta} \qquad (|):\ \frac{P : \Gamma \vdash \Delta \quad Q : \Gamma \vdash \Delta}{P \mid Q : \Gamma \vdash \Delta}$$

$$(\mathit{let}):\ \frac{P : \Gamma, y{:}B \vdash x{:}A, \Delta}{\mathit{let}\,\langle x,y\rangle = z\ \mathit{in}\ P : \Gamma, z{:}A \rightarrow B \vdash \Delta}\ (y,z \notin \Delta;\ x \notin \Gamma) \qquad (\langle\rangle\text{-}out):\ \frac{P : \Gamma, b{:}A \vdash c{:}B, \Delta}{\overline{a}\langle b,c\rangle.P : \Gamma, b{:}A \vdash a{:}A \rightarrow B, c{:}B, \Delta}\ (a \notin \Delta;\ a,c \notin \Gamma)$$

We write $P : \Gamma \vdash^{io}_{\pi} \Delta$ if there exists a derivation using these rules that has this expression in the conclusion.

This is the system as first defined in [7] and is also used in [10]. Notice that the '*input-output interface of a π-process*' property is nicely preserved by all the rules; handling of arrow types is restricted by the type system to the rules (let) and $(\langle\rangle\text{-}out)$.

Lemma 2.9 *The inference rules*

$$(Weak):\ \frac{P : \Gamma \vdash \Delta}{P : \Gamma' \vdash \Delta'}\ (\Gamma' \supseteq \Gamma, \Delta' \supseteq \Delta) \qquad (\langle\rangle\text{-}in):\ \frac{P : \Gamma, y{:}B \vdash x{:}A, \Delta}{a(x,y).P : \Gamma, a{:}A \rightarrow B \vdash \Delta}\ (y,a \notin \Delta,\ x \notin \Gamma)$$

$$(out'):\ \frac{}{\overline{a}\,b : \Gamma, b{:}A \vdash a{:}A, b{:}A, \Delta} \qquad (\langle\rangle\text{-}out'):\ \frac{}{\overline{a}\langle b,c\rangle : \Gamma, b{:}A \vdash a{:}A \rightarrow B, c{:}B, \Delta}$$

$$(\rightarrow):\ \frac{}{!\,u \rightarrow a : u{:}A \vdash a{:}A} \qquad (\overline{\nu}):\ \frac{\llbracket Q \rrbracket\, w : \Gamma \vdash w{:}B, \Delta}{\overline{b}(w).\llbracket Q \rrbracket\, w : \Gamma \vdash b{:}B, \Delta}$$

are admissible.

Proof: That weakening is admissible follows by a straightforward reasoning over the structure of derivations; for the other rules, consider:

$$\frac{\dfrac{\boxed{}}{P : \Gamma, y{:}B \vdash^{io}_{\pi} x{:}A, \Delta}}{\mathit{let}\,\langle x,y\rangle = z\ \mathit{in}\ P : \Gamma, z{:}A \rightarrow B \vdash^{io}_{\pi} \Delta}\ (\mathit{let})}{a(z).\mathit{let}\,\langle x,y\rangle = z\ \mathit{in}\ P : \Gamma, a{:}A \rightarrow B \vdash^{io}_{\pi} \Delta}\ (in)$$

[9] Type assignment is classical in nature (*i.e.* not intuitionistic), since we can have more than one conclusion.

$$\dfrac{\dfrac{}{\mathbf{0} : \Gamma, b{:}A \vdash^{io}_{\pi} b{:}A, \Delta}\,(\mathbf{0})}{\overline{a}\,b.\mathbf{0} : \Gamma, b{:}A \vdash^{io}_{\pi} a{:}A, b{:}A, \Delta}\,(out) \qquad \dfrac{\dfrac{}{\mathbf{0} : \Gamma, b{:}A \vdash^{io}_{\pi} c{:}B, \Delta}\,(\mathbf{0})}{\overline{a}\langle b,c\rangle.\mathbf{0} : \Gamma, b{:}A \vdash^{io}_{\pi} a{:}A \to B, c{:}B, \Delta}\,(\langle\rangle\text{-}out)$$

$$\dfrac{\dfrac{}{\overline{a}\,w : \Gamma, w{:}A \vdash^{io}_{\pi} a{:}A, w{:}A}\,(out')}{u(w).\overline{a}\,w : \Gamma, u{:}A \vdash^{io}_{\pi} a{:}A}\,(in) \qquad \dfrac{\dfrac{\dfrac{\boxed{}}{\llbracket Q \rrbracket\, w : \Gamma \vdash^{io}_{\pi} w{:}B, \Delta}}{\overline{b}\,w.\llbracket Q \rrbracket\, w : \Gamma \vdash^{io}_{\pi} b{:}B, w{:}B, \Delta}\,(out)}{(\nu w)\,(\overline{b}\,w.\llbracket Q \rrbracket\, w) : \Gamma \vdash^{io}_{\pi} b{:}B, \Delta}\,(\nu)$$

□

This notion is a true assignment system, which does not (directly) relate back to any logic. For example, rules $(|)$ and $(!)$ do not change the left and right contexts, so do not correspond to any logical rule. Moreover, rule (ν) just removes a formula, and rule $(\langle\rangle\text{-}out)$ is clearly not an instance of an axiom. The only known relation this system has with logic is that established in [7, 8].

Since weakening is included, we allow ourselves to be a little less precise when we construct derivations, and freely switch to multiplicative style where rules join contexts, by using, for example, the rule

$$(|):\ \frac{P_1 : \Gamma_1 \vdash \Delta_1 \quad \cdots \quad P_n : \Gamma_n \vdash \Delta_n}{P_1 \mid \cdots \mid P_n : \Gamma_1, \ldots, \Gamma_n \vdash \Delta_1, \ldots, \Delta_n}$$

whenever convenient.

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 [37]; Milner's interpretation is input based and the interpretation of closed λ-terms respects large-step *lazy* reduction $\to_{\mathrm{L}}$ [3] to normal form up to substitution; this was later generalised to β-equality, but using weak bisimilarity [44].

For many years, it seemed that the first and final word on the interpretation of the λ-calculus has been said by Milner; 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 [44] constitute examples, also in the context of the higher-order π-calculus; [30] used Milner's approach with a typed version of the π-calculus; [47] used it in the context of continuation-passing style languages.

It is defined by:

Definition 3.1 (Milner's interpretation [37]) Let a not be a λ-variable. Then

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

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

Notice that, in $\llbracket MN \rrbracket^{\mathrm{M}}\, a$, the interpretation of the operand N is placed under output, and thereby blocked from running; this comes at a price: now β-reductions that occur in the

operand 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*.

After Milner's original encoding, many variants followed; for example, [44] defines an encoding that respects lazy reduction. We repeat that definition here, but adjusted to the normal π-calculus, rather than the higher-order one:

Definition 3.2 (*cf.* [44]) The encoding $[\![\cdot]\!]^{\mathcal{N}}_{\rm l}\cdot$[10] of the lazy λ-calculus is defined through:

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

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

For this encoding, [44] shows a number of results: first it shows:

$$[\![(\lambda x.M)N]\!]^{\mathcal{N}}_{\rm l}\, p \overset{(1)}{\xrightarrow{\tau}}{}^{2}_{d}\ (\nu x)\,([\![M]\!]^{\mathcal{N}}_{\rm l}\, p \mid !\,x(w).[\![N]\!]^{\mathcal{N}}_{\rm l}\, w) \overset{(2)}{\approx}_{g}\ [\![M[N/x]]\!]^{\mathcal{N}}_{\rm l}\, p$$

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

$$M \rightarrow_{\rm L} N \overset{(3)}{\Rightarrow} [\![M]\!]^{\mathcal{N}}_{\rm l}\, p \xrightarrow{\tau}{}^{2}_{d}\!\approx_g [\![N]\!]^{\mathcal{N}}_{\rm l}\, p$$

and[11]

$$M =_\beta N \overset{(4)}{\Rightarrow} [\![M]\!]^{\mathcal{N}}_{\rm l}\, p \approx_g [\![N]\!]^{\mathcal{N}}_{\rm l}\, p$$

We show the equivalent of these results for our encoding in Theorem 7.1 and Theorem 7.6 below.

The characterisation of $\llbracket M \rrbracket^{\rm M}_{\rm l}\, a \approx \llbracket N \rrbracket^{\rm M}_{\rm l}\, a$, left as open problem in [37], was achieved through showing that

$$\llbracket M \rrbracket^{\rm M}_{\rm l}\, a \approx \llbracket N \rrbracket^{\rm M}_{\rm l}\, a \text{ if and only if } M \cong N,$$

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

However, applicative bisimulation equates the terms $x(x\Theta\Delta\Delta)\Theta$ 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 $\llbracket\cdot\rrbracket^{\rm M}_{\rm l}\cdot$. 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 and by exploiting a more abstract encoding into the *Higher Order* π-calculus, a variant of the π-calculus with higher-order communications. Sangiorgi's result then essentially states that the interpretations of closed Λ_c-terms M and N are contextually equivalent if and only if M and N are applicatively bisimilar; in [43] he improves on this by showing that the interpretation

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

[11] In [44], 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.

of terms in Λ_c in the standard π-calculus is weakly bisimilar if and only if they have the same Lévy-Longo tree [35, 36] (a lazy variant of Böhm trees [14]).

An output-based encoding for the λ-calculus

In [10] we presented a *logical*, *output-based* interpretation $\llbracket \cdot \rrbracket^{s}_{a}$ 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:

Definition 3.3 (Spine interpretation [10]) Let a not be a λ-variable. Then

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

For this interpretation, [10] 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 5.2, and the notion of type assignment defined in Definition 2.8. The main results shown are:

Theorem 3.4 ([10]) *i) If $M\Uparrow$ then $\llbracket M \rrbracket^{s}_{a}\Uparrow$, and if $M \rightarrow_{\mathrm{xH}} N$ then $\llbracket M \rrbracket^{s}_{a} \rightarrow^{*}_{\pi} \sim_{\mathrm{C}} \llbracket N \rrbracket^{\mathrm{H}}_{a}$.*
ii) If $\Gamma \vdash M : A$ then $\llbracket M \rrbracket^{s}_{a} : \Gamma \vdash^{io}_{\pi} a{:}A$.

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

As argued in [10], to show the above result, which formulates a direct *step-by-step* relation between β-reduction and the synchronisation in the π-calculus, it is necessary to make the substitution explicit. This is a direct consequence of the fact that, in the π-calculus, the implicit substitution of the λ-calculus gets 'implemented' *one variable at the time*, rather than all in one fell swoop. Since we aim to show a similar result for $\lambda\mu$, we will therefore also here define a notion of explicit substitution.

Classical logic and the π-calculus

A natural extension of this line of research is to see if the π-calculus can be used to interpret more complex calculi as well, as for example calculi that relate not to intuitionistic logic, but to classical logic, as $\lambda\mu$, $\lambda\mu\tilde{\mu}$, or $\mathcal{X}$. There are, to date, a number of papers on this topic.

In [30] 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 full typed terms and processes only. The syntax of processes there considered is

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

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

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

as the basic synchronisation rule, as well as

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

where $\mathtt{C}[\cdot]$ 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. Types for processes prescribe usage of names, and name passing is restricted to *bound (private, hidden) name passing*.[12]

On the relation between Girard's linear logic [25] and the π-calculus, [16] gives a treatment of information flow in proof-nets; only a small fragment of Linear Logic was considered, and the translation between proofs and π-calculus was left rather implicit, as also noted in [20].

To illustrate this, notice that [16] uses the standard syntax for the polyadic π-calculus

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

similar to the one we use here (see Definition 2.1) but for the fact that in [16] the *let*-construct is not used. However, the encoding of a 'cut' in linear logic

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

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:

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

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, 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 [16] becomes:

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

Although intended, no communication is possible in this term. We have 'underbraced' the desired communication which is impossible, as the arity of the channel k does not match. To overcome this kind of problem, Bellin and Scott would need the *let*-construct with use of pairs of names as we have introduced in this paper in Definition 2.1. Moreover, there is no relation between the interpreted terms and proofs stated in [16] in terms of logic, types, or provable statements; here, we make a clear link between interpreted proofs and the logic through our notion of type assignment for the π-calculus.

In [7] an interpretation into π of the sequent calculus $\mathcal{X}$ is defined that enjoys the Curry-Howard isomorphism for Gentzen's LK [24], which is shown to respect reduction. However, this result is only partial, as it is formulated as "*if* $P \rightarrow_{\mathcal{X}} Q$, *then* $[\![P]\!] \sqsubseteq_{c} [\![Q]\!]$", allowing $[\![P]\!]$ to have more observable behaviour than $[\![Q]\!]$; the main reason for this is that reduction in $\mathcal{X}$ is non-confluent. Although in [7] it is reasoned that this is natural in the context of non-confluent, symmetric sequent calculi, and there it is shown that the interpretation preserves types, it is a weaker result than could perhaps be desired or expected.

An interpretation of $\lambda\mu\tilde{\mu}$ is studied in [22]; the interpretation defined there strongly depends

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

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 [10] is that it establishes a strong link between reduction in the π-calculus and step-by-step *explicit substitution* [17] 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 [37] with respect to explicit lazy reduction, all defined in [10]. In view of this, for the purpose of defining an interpretation for $\Lambda\mu$ into the π-calculus in [11], it was natural to study a variant of $\Lambda\mu$ with explicit substitution as well; since here we work with $\lambda\mu$, we present here $\lambda\mu\mathbf{x}$, as a variant of $\Lambda\mu\mathbf{x}$ as presented in that paper.

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 *explicit $\lambda\mu$ calculus*, $\lambda\mu\mathbf{x}$, is defined by:

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

We consider the occurrences of x in M bound in $M\langle x := N\rangle$, and those of α in M in $M\langle \alpha := N\cdot\gamma\rangle$; by Barendregt's convention, x and α do not appear outside M.

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

a) Main reduction rules:[13]

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

b) Term substitution rules:

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

c) Structural rules:[14]

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

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

[14] Notice that these rules are, in part, defined on pseudo-terms.

d) Contextual rules:

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

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

Notice that we do not allow explicit substitutions to 'cross' and do not add rules like

$$M\langle x := N\rangle\langle y := L\rangle \ \to \ M\langle y := L\rangle\langle x := N\langle y := L\rangle\rangle$$

or

$$M\langle x := N\rangle\langle y := L\rangle \ \to \ M\langle y := L\rangle\langle x := N\rangle\langle y := L\rangle$$

As in [17], this would introduce undesired non-termination. We will add the latter for our notion of explicit head reduction in Section 5, but will limit its applicability, thereby avoiding the problem. This notion differs from that of [6], where a version with explicit substitution is defined for a variant of $\lambda\mu$ that uses de Bruijn indices [19]. Notice that since reduction in $\lambda\mu\mathbf{x}$ actually is formulated via term rewriting rules [32], reduction is allowed to take place also *inside the substitution term*.

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

Proposition 4.2 ($\lambda\mu\mathbf{x}$ IMPLEMENTS $\lambda\mu$-REDUCTION) *i)* $M \to_{\beta\mu} N \Rightarrow M \to_{\mathbf{x}}^{*} N$.

ii) $M \in \lambda\mu \ \& \ M \to_{\mathbf{x}} N \Rightarrow \exists L \in \lambda\mu \ [N \to_{:=}^{*} L]$.

The notion of type assignment on $\lambda\mu\mathbf{x}$ is a natural extension of the system for the $\lambda\mu$-calculus of Definition 1.2 by adding rules (*T-cut*) and (*C-cut*).

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:

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

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

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

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

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 [33]) or perform a contextual substitution only on names that occur in front of the term. The results of [10] show that this is exactly the kind of reduction that the π-calculus naturally encodes.

We first formally define the following notions:

Definition 5.1 (Head variables and head names) For terms of $\lambda\mu\mathbf{x}$, the *head variable* of M, $hv(M)$, and the *head name* $hn(M)$ are defined by:

$$\begin{array}{lcll}
hv(\lambda x.M) & = & hv(M) & \\
hv(xM_1\cdots M_n) & = & x & \\
hv(\mu\alpha.[\beta]M) & = & hv(M) & \\
hv(M\langle x:=N\rangle) & = & hv(M) & (hv(M)\neq x)\\
hv(M\langle \alpha:=N\cdot\gamma\rangle) & = & hv(M) & \\
 & & & \\
hn(\mu\alpha.[\beta]\boldsymbol{H}) & = & \beta & \\
hn(M\langle x:=N\rangle) & = & hn(M) & \\
hn(M\langle \alpha:=N\cdot\gamma\rangle) & = & hn(M) & (hn(M)\neq \alpha)
\end{array}$$

We let $x = hv(M)$ be true whenever $hv(M)$ is defined and returns x, and false if either it is undefined or does not return x. We write $hn(M)\in V$ whenever $hn(M)=\alpha$ (so $hn(M)$ is defined) and $\alpha\in V$, and $hn(M)\notin V$ when this is not the case (so either $hn(M)$ is not defined, or $hn(M)=\beta$, and $\beta\notin V$).

Notice that, for example, $hv((\lambda x.M)N)$, $hv(\lambda y.x\langle x:=N\rangle)$, $hn(\lambda x.M)$, and $hn(PQ)$ are undefined.

This leads to the definition of the following:

Definition 5.2 (Explicit head-reduction) We define *explicit head-reduction* $\to_{\mathrm{xH}}$ on $\lambda\mu\mathbf{x}$ as $\to_{\mathrm{x}}$, but change and add a few rules:

i) The main reduction rules are as before:

$$\begin{array}{rcll}
(\lambda x.M)N & \to & M\langle x:=N\rangle & (\beta\text{-}rule)\\
(\mu\alpha.\mathsf{C})N & \to & \mu\gamma.\mathsf{C}\langle\alpha:=N\cdot\gamma\rangle & (\gamma\ fresh)\\
\mu\alpha.[\alpha]M & \to & M & (\alpha\notin fn(M))\\
\mu\alpha.[\beta]\mu\gamma.\mathsf{C} & \to & \mu\alpha.\mathsf{C}[\beta/\gamma] &
\end{array}$$

ii) In the term substitution rules, we replace the rule for application and add side-conditions:

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

iii) There are only two structural rules:

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

iv) We only allow the following contextual rules:

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

v) We add one substitution rules:

$$M\langle x := N\rangle\langle y := P\rangle \ \to \ M\langle y := P\rangle\langle x := N\rangle\langle y := P\rangle \quad (y = hv(M))$$

Notice that, for example, in case (*ii*), the fourth of the clauses postpones the substitution $\langle x := N\rangle$ on Q until such time that an occurrence of the variable x in Q becomes the head-variable of the full term, and that, without the side-condition, we would allow

$$(yx)\langle x := N\rangle \ \to \ (y\langle x := N\rangle x)\langle x := N\rangle \ \to \ (yx)\langle x := N\rangle$$

which is obviously undesirable.

Remark that we need to add the rules of case (*v*): if we take the term $P\langle x := Q\rangle\langle y := R\rangle$, under 'normal' reduction $\to_{\mathbf{x}}$, the innermost substitution has to complete first before the outermost can run. When moving towards explicit head-reduction, this would mean that we cannot reduce, for example, $yx\langle x := Q\rangle\langle y := R\rangle$; the innermost substitution cannot advance, since x is not the head variable, and the outermost cannot advance since it is not defined for the substitution term it has to work on, $yx\langle x := Q\rangle$ in this case. To allow outermost substitution to 'jump' the innermost, we need to add the extra rule. As mentioned above, this potentially introduces non-termination, but by demanding that the variable concerned is actually the head-variable of the term, we avoid this. Notice that the outermost substitution $\langle y := N\rangle$ in $M\langle y := L\rangle\langle x := N\rangle\langle y := L\rangle$ cannot propagate inside, since y is not the head variable of $M\langle y := L\rangle$, even if it is that of M (see Definition 5.3). Notice that we violate Barendregt's convention by the rules in case (*v*), and assume that the occurrences of y in M are bound by the innermost substitution, and those in N by the outermost. We could avoid this by introducing renaming, as usual, by writing

$$M\langle x := N\rangle\langle y := P\rangle \ \to \ M[z/y]\langle z := P\rangle\langle x := N\rangle\langle y := P\rangle \quad (y = hv(M), z \textit{ fresh})$$

but will not do that here.

Definition 5.3 The normal forms with respect to $\to_{\mathbf{xH}}$ are defined through the grammar:

$$\begin{array}{rcll} \boldsymbol{N} & ::= & \lambda x.\boldsymbol{N} & \\ & | & xM_1\cdots M_n & (n \geq 0) \\ & | & \mu\alpha.[\beta]\boldsymbol{N} & (\alpha \neq \beta \vee \alpha \in \boldsymbol{N}, \boldsymbol{N} \neq \mu\gamma.[\delta]\boldsymbol{N}') \\ & | & \boldsymbol{N}\langle x := M\rangle & (hv(\boldsymbol{N}) \neq x) \\ & | & \boldsymbol{N}\langle \alpha := M\cdot\gamma\rangle & (hn(\boldsymbol{N}) \neq \alpha) \end{array}$$

It is straightforward to check that these terms are indeed the normal forms with respect to $\to_{\mathbf{xH}}$.

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

Proposition 5.4 *i) If $M \to_{\mathrm{H}}^{*} N$, then there exists $L \in \lambda\mu\mathbf{x}$ such that $M \to_{\mathbf{xH}}^{*} L$ and $L \to_{:=}^{*} N$.*

ii) If $M \to_{\mathbf{xH}}^{nf} N$ with $M \in \lambda\mu$, then there exists $L \in \lambda\mu$ such that $N \to_{:=}^{nf} L$, and $M \to_{\mathrm{H}}^{nf} L$.

iii) $M \to_{\beta\mu}^{nf} N$ if and only if there exists $L \in \lambda\mu\mathbf{x}$ such that $M \to_{\mathbf{xH}}^{nf} L$ and $L \to_{\mathbf{x}}^{*} N$.

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

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

Example 5.5 *i)*

$$\begin{array}{lllll}
(\lambda x.xx)(\lambda y.y) & \to_{\mathbf{x}\text{H}} & xx\,\langle x:=\lambda y.y\rangle & \to_{\mathbf{x}\text{H}} & \\
(x\,\langle x:=\lambda y.y\rangle x)\,\langle x:=\lambda y.y\rangle & \to_{\mathbf{x}\text{H}} & (\lambda y.y)x\,\langle x:=\lambda y.y\rangle & \to_{\mathbf{x}\text{H}} & \\
y\,\langle y:=x\rangle\,\langle x:=\lambda y.y\rangle & \to_{\mathbf{x}\text{H}} & x\,\langle x:=\lambda y.y\rangle & \to_{\mathbf{x}\text{H}} & \lambda y.y
\end{array}$$

Notice that this reduction is deterministic.

ii) Reduction in $\to_{\mathbf{x}\text{H}}$ is not deterministic in general:

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

iii)

$$\begin{array}{lllll}
(\mu\alpha.[\beta]\mu\delta.[\alpha](\lambda y.y))(\lambda z.z) & \to_{\mathbf{x}\text{H}} & (\mu\gamma.[\beta]\mu\delta.[\alpha](\lambda y.y))\,\langle\alpha:=\lambda z.z\cdot\gamma\rangle & \to_{\mathbf{x}\text{H}} & \\
(\mu\gamma.[\alpha](\lambda y.y)[\beta/\delta])\,\langle\alpha:=\lambda z.z\cdot\gamma\rangle & = & (\mu\gamma.[\alpha](\lambda y.y))\,\langle\alpha:=\lambda z.z\cdot\gamma\rangle & \to_{\mathbf{x}\text{H}} & \\
\mu\gamma.[\gamma]((\lambda y.y)\,\langle\alpha:=\lambda z.z\cdot\gamma\rangle)(\lambda z.z) & \to_{\mathbf{x}\text{H}} & \mu\gamma.[\gamma](\lambda y.y)(\lambda z.z) & \to_{\mathbf{x}\text{H}} & \\
\mu\gamma.[\gamma]y\,\langle y:=\lambda z.z\rangle & \to_{\mathbf{x}\text{H}} & \mu\gamma.[\gamma]\lambda z.z & \to_{\mathbf{x}\text{H}} & \lambda z.z
\end{array}$$

Notice that this reduction sequence is not deterministic.

iv) Some reductions leave substitutions in place:

$$\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)) \;\to_{\mathbf{x}\text{H}}\; \lambda f.(f(xx)\,\langle x:=\lambda x.f(xx)\rangle)$$

and the last term is in $\to_{\mathbf{x}\text{H}}$-normal form.

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

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

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

$$xy\,\langle y:=x\rangle\,\langle x:=\Delta\rangle \;\to^*_{:=}\; xx\,\langle x:=\Delta\rangle \;\to^*_{:=}\; \Delta\Delta$$

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

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

We will now define our logical,[15] output-based interpretation $\llbracket M \rrbracket\, a$ of the $\lambda\mu\mathbf{x}$-calculus into the π-calculus (where M is a $\lambda\mu$-term, and a is the name given to its (anonymous) output),

[15] It is called *logical* because it has its foundation in the relation between natural deduction and Gentzen's sequent calculus LK; in particular, the case for application is based on the representation of *modus ponens*

$$\frac{\Gamma\vdash A\Rightarrow B \quad \Gamma\vdash A}{\Gamma\vdash B} \quad \text{by} \quad \frac{\Gamma\vdash A\Rightarrow B \qquad \dfrac{\Gamma\vdash A \quad \Gamma,B\vdash B}{\Gamma,A\Rightarrow B\vdash B}}{\Gamma\vdash B}$$

which is essentially the one presented in [11], but no longer considers $[\alpha]M$ to be a term.

The main idea behind the interpretation, as in [10], 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. [11])) Let a not be a $\lambda\mu$-variable or name. Then

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

Notice that not all the interpreted entities on the left are actual $\lambda\mu$ terms; moreover,

$$\llbracket \mu\gamma.[\beta]M \rrbracket\, a \;\triangleq\; (\nu s)\,\llbracket [\beta]M \rrbracket\, s\,[a/\gamma] \;\triangleq\; (\nu s)\,\llbracket M \rrbracket\, \beta\,[a/\gamma] \;\equiv\; \llbracket M \rrbracket\, \beta\,[a/\gamma]$$

which implies that we could have added

$$\llbracket \mu\gamma.[\beta]M \rrbracket\, a \;\triangleq\; \llbracket M \rrbracket\, \beta\,[a/\gamma]$$

to our encoding. However, treating the naming and μ-binding separately is convenient later in the proofs.

The interpretation presented in [11] had the case

$$\llbracket \mu\gamma.M \rrbracket\, a \;\triangleq\; (\nu s)\,\llbracket M \rrbracket\, s\,[a/\gamma] \quad (s \textit{ fresh})$$

so was defined for $\Lambda\mu$; note that the this encoding very elegantly expresses that the main computation in $\mu\gamma.M$ is blocked: the name s is fresh and bound, so the main output of $(\nu s)\,\llbracket M \rrbracket\, s\,[a/\gamma]$ 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, using that interpretation, the process

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

is in normal form. Notice that all inputs and outputs are restricted; thereby, this process is weakly bisimilar to $\mathit{0}$ and to $\llbracket \Delta\Delta \rrbracket\, a$ (see Lemma 8.8). So using that interpretation, we cannot distinguish between *blocked* and *looping* computations, which clearly would affect any full-abstraction result. When restricting our interpretation to $\lambda\mu$, this problem disappears: since naming has to follow μ-abstraction, $\mu\alpha.\lambda x.x$ is not a term in $\lambda\mu$; instead, now

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

which outputs on β.

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

$$\llbracket \mu\gamma.\mathsf{C} \rrbracket\, a \;\triangleq\; (\nu s)\,(\llbracket \mathsf{C} \rrbracket\, s \mid !\,\gamma \rightarrow a)$$

which is operationally the same as $(\nu s)\,\llbracket \mathsf{C} \rrbracket\, s\,[a/\gamma]$ (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 7.7), a property that is of use in the proof of termination (Theorem 7.8).

As in [12], we can make the following observations:

Remark 6.2

- The synchronisations generated by the encoding only involve processes of the shape:

$$x(w).\overline{\alpha}\,w \qquad \overline{\beta}\langle x,\alpha\rangle \qquad z(\beta,y).(P \mid Q)$$

  so in particular, substitution is always well defined. These synchronisations are of the shape:

$$(\nu c)\,((\nu yb)\,(P \mid \overline{c}\langle y,b\rangle) \mid c(v,d).(R \mid d(w).\overline{a}\,w)) \;\rightarrow_{\pi}\; (\nu yb)\,(P \mid R[y/v] \mid b(w).\overline{a}\,w)$$

  and after the synchronisation over c, P can receive over y from $R[y/\alpha]$ and send over b to $b(w).\overline{a}\,w$; or of the shape

$$(\nu c)\,((\nu yb)\,(P \mid \overline{c}\langle y,b\rangle) \mid c(w).\overline{a}\,w) \;\rightarrow_{\pi}\; (\nu yb)\,(P \mid \overline{a}\langle y,b\rangle)$$

- All synchronisation takes place *only* over channels whose names are bound names or variables in the terms that are interpreted.
- To underline the significance of our results, notice that the encoding is not trivial, since

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

  processes that differ under $\approx$.

Notice that, as is the case for Milner's interpretation and in contrast to the interpretation of [10], a guard is placed on the replicated terms. This is not only done with an eye on proving preservation of termination, but more importantly, to make sure that $(\nu x)\,(\llbracket x{:=}N \rrbracket) \approx \mathbf{0}$, a property we need for our full abstraction result: since a term can have named sub-terms, the interpretation will generate output not only for the term itself, but also for those named terms, so the process $(\nu x)\,(!\llbracket N \rrbracket\, x)$ – using the variant of [10] – *can have* observable behaviour, in contrast to here, where $(\nu x)\,(!\,\overline{x}(w).\llbracket N \rrbracket\, w)$ *is* weakly bismilar to $\mathbf{0}$.

In [10] the case for application in the interpretation for λ-terms was defined as:

$$\llbracket MN \rrbracket^{\mathrm{s}}_{\mathrm{H}}\, a \;\triangleq\; (\nu c)\,(\llbracket M \rrbracket^{\mathrm{H}}_{\mathrm{H}}\, c \mid c(v,d).(!\llbracket N \rrbracket^{\mathrm{H}}_{\mathrm{H}}\, v \mid d{\rightarrow}a))$$

In particular, there the input on name c is *not replicated*: this corresponds to the fact that for λ-terms, in $\llbracket M \rrbracket^{\mathrm{H}}_{\mathrm{H}}\, c$, 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 $\llbracket \mu\alpha.[\alpha]M \rrbracket\, a = \llbracket M \rrbracket\, \alpha\,[a/\alpha] = \llbracket M[a/\alpha] \rrbracket\, a$, then the name a appears many times in the latter.

Remark 6.3 Observe the similarity between

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

The first communicates N via the output channel c of M (which might occur more than once inside $\llbracket M \rrbracket\, c$, so replication is needed), whereas the second communicates with all the sub-terms that have c as output name, and changes the output name of the process to γ.[16] In other

[16] A similar observation can be made for the interpretation of $\lambda\mu$ in $\mathcal{X}$ [9].

words, explicit structural substitution is just a special case of application. As an abbreviation, we will write $(\nu c)\,(\llbracket M \rrbracket\, c \mid \llbracket c := N{\cdot}a \rrbracket)$ for $\llbracket MN \rrbracket\, a$.

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 order to be able to express it; the component of our interpretation that deals with pure λ-terms is almost exactly that of [10] (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 structural substitution is dealt with entirely by congruence (see also Example 6.7).

This strengthens our view that, as far as our interpretation is concerned, μ-reduction is not a separate computational step, but is essentially 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 iin the π-calculus involves a computational step, but context switches are dealt with by congruence; this is only possible, of course, because the interpretation of the operand in application uses replication. This stresses that the π-calculus constitutes a very powerful abstract machine indeed.

We would like to stress that, although inspired by logic, our interpretation does not depend on types *at all*; in fact, we can treat untypeable terms as well, and can show that $\llbracket \Delta\Delta \rrbracket\, a$ (perhaps the prototype of a non-typeable term) runs forever without generating output (see Example 8.1; this already holds for the interpretation of [10]).

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. First we need to show:

Proposition 6.4 ([12]) $\quad (\nu xb)\,(c(v,d).(P \mid !d{\to}e)) \;\approx\; (\nu a)\,(!a{\to}e \mid (\nu xb)\,(c(v,d).(P \mid !d{\to}a)))$

We use this result to show the following:

Lemma 6.5 (Renaming lemma) *Let $a \notin fv\,(M)$, and $a \neq e$, then*

i) $(\nu a)\,(!a{\to}e \mid \llbracket M \rrbracket\, a) \;\approx\; \llbracket M \rrbracket\, e.$

ii) $(\nu a)\,(!a{\to}e \mid \llbracket M \rrbracket\, b) \;\approx\; \llbracket M[e/a] \rrbracket\, b \quad (b \neq a).$

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

$$(M = x):\; (\nu a)\,(!a{\to}e \mid \llbracket x \rrbracket\, a) \;\triangleq\; (\nu a)\,(!a{\to}e \mid x(u).!u{\to}a) \;\approx\; x(u).!u{\to}e \;\triangleq\; \llbracket x \rrbracket\, e$$

$$\begin{aligned}(M = \lambda x.N):\;\; & (\nu a)\,(!a{\to}e \mid \llbracket \lambda x.N \rrbracket\, a) && \triangleq\; (\nu a)\,(!a{\to}e \mid (\nu xb)(\llbracket N \rrbracket\, b \mid \overline{a}\langle x,b\rangle)) \;\to_\pi (a)\\ & (\nu axb)\,(!a{\to}e \mid \llbracket N \rrbracket\, b \mid \overline{e}\langle x,b\rangle) && \approx\; (a \neq x \Rightarrow a \notin fv\,(N))\\ & (\nu a)\,(!a{\to}e) \mid (\nu xb)(\llbracket N \rrbracket\, b \mid \overline{e}\langle x,b\rangle) && \approx\; \llbracket \lambda x.N \rrbracket\, e\end{aligned}$$

$$\begin{aligned}(M = PQ):\;\; & (\nu a)\,(!a{\to}e \mid \llbracket PQ \rrbracket\, a) && \triangleq\\ & (\nu a)\,(!a{\to}e \mid (\nu c)\,(\llbracket P \rrbracket\, c \mid !c(v,d).(!\overline{v}(w).\llbracket Q \rrbracket\, w \mid !d{\to}a))) && \approx\; (6.4, a \neq c, a \notin fv\,(P,Q))\\ & (\nu c)\,(\llbracket P \rrbracket\, c \mid !c(v,d).(!\overline{v}(w).\llbracket Q \rrbracket\, w \mid !d{\to}e)) && \triangleq\; \llbracket PQ \rrbracket\, e\end{aligned}$$

$$\begin{aligned}(M = P\langle x{:=}Q\rangle):\;\; & (\nu a)\,(!a{\to}e \mid \llbracket P\langle x{:=}Q\rangle \rrbracket\, a) && \triangleq\\ & (\nu a)\,(!a{\to}e \mid (\nu x)(\llbracket P \rrbracket\, a \mid !\overline{x}(w).\llbracket Q \rrbracket\, w)) && \approx\; (IH(i), x \neq a, a \notin fv\,(P,Q))\\ & (\nu x)(\llbracket P \rrbracket\, e \mid !\overline{x}(w).\llbracket Q \rrbracket\, w) && \triangleq\; \llbracket P\langle x{:=}Q\rangle \rrbracket\, e\end{aligned}$$

$$
\begin{array}{lll}
(M = \mu\beta.[\gamma]N): \quad (\nu a)\,(!\,a{\rightarrow}e \mid \llbracket \mu\beta.[\gamma]N \rrbracket\, a) & \triangleq & \\
\quad (\nu a)\,(!\,a{\rightarrow}e \mid \llbracket N \rrbracket\, \gamma[a/\beta]) & \approx & (IH(ii), a \neq \gamma, a \notin fv(N)) \\
\quad \llbracket N \rrbracket\, \gamma[a/\beta]\,[e/a] & \approx & \llbracket N \rrbracket\, \gamma[e/\beta] \;\triangleq\; \llbracket \mu\beta.[\gamma]N \rrbracket\, e \\
(M = P\,\langle \beta{:=}Q{\cdot}\gamma \rangle): \quad (\nu a)\,(!\,a{\rightarrow}e \mid \llbracket P\,\langle \beta{:=}Q{\cdot}\gamma \rangle \rrbracket\, a) & \triangleq & \\
\quad (\nu a)\,(!\,a{\rightarrow}e \mid (\nu\beta)(\llbracket P \rrbracket\, a \mid \llbracket \beta{:=}Q{\cdot}\gamma \rrbracket)) & \approx & (IH(i), a \neq \beta, a \notin fv(Q)) \\
\quad (\nu\beta)(\llbracket P[e/a] \rrbracket\, e \mid \llbracket \beta{:=}Q[e/a]{\cdot}\gamma \rrbracket) & \triangleq & (a \notin fv(P)) \\
\quad (\nu\beta)(\llbracket P \rrbracket\, e \mid \llbracket \beta{:=}Q{\cdot}\gamma \rrbracket) & \triangleq & \llbracket P\,\langle \beta{:=}Q{\cdot}\gamma \rangle \rrbracket\, e
\end{array}
$$

□

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

Definition 6.6 *i*) We define a *garbage collection* bisimilarity by: $P \approx_{\mathrm{G}} Q$ if and only if there exists R such that $P = Q \mid R$ and $R \approx 0$. We call a process that is weakly bisimilar to 0 *garbage*.

ii) We define $\approx_{\mathrm{R}}$ as the largest, symmetric equivalence such that:

a) for all P such that the name a is a free output of P, is different from e, and is only used to output: $(\nu a)\,(P \mid !\,a{\rightarrow}e) \approx_{\mathrm{R}} P[e/a]$,

b) for all contexts C, if $P \approx_{\mathrm{R}} Q$ then $\mathsf{C}[P] \approx_{\mathrm{R}} \mathsf{C}[Q]$.

iii) We define $\approx\!\!\rightarrow^*_\pi$ as $\rightarrow^*_\pi \cup \approx_{\mathrm{R}} \cup \approx_{\mathrm{G}}$.

So $\approx_{\mathrm{R}}$ is used when we want to emphasise that two processes are equivalent just using renaming. Notice that $\approx_{\mathrm{G}} \subset \approx$ and $\approx_{\mathrm{R}} \subset \approx$.

Using this lemma, we can show the following:

Example 6.7 The interpretation of the β-redex $(\lambda x.P)Q$ reduces as follows:

$$
\begin{array}{lll}
\llbracket (\lambda x.P)Q \rrbracket\, a & \triangleq & \\
(\nu c)\,((\nu xb)(\llbracket P \rrbracket\, b \mid \overline{c}\langle x,b \rangle) \mid !\,c(v,d).(\llbracket v{:=}Q \rrbracket \mid !\,d{\rightarrow}a)) & \rightarrow_\pi & (c) \\
(\nu c)\,((\nu bx)(\llbracket P \rrbracket\, b \mid !\,b{\rightarrow}a \mid \llbracket x := Q \rrbracket) \mid !\,c(v,d).(\llbracket v{:=}Q \rrbracket \mid !\,d{\rightarrow}a)) & \equiv & (c \notin fn(P,Q)) \\
(\nu bx)\,(\llbracket P \rrbracket\, b \mid !\,b{\rightarrow}a \mid \llbracket x{:=}Q \rrbracket) \mid (\nu c)\,(\llbracket c{:=}Q{\cdot}a \rrbracket) & \approx_{\mathrm{G}} & (*) \\
(\nu bx)\,(\llbracket P \rrbracket\, b \mid !\,b{\rightarrow}a \mid \llbracket x{:=}Q \rrbracket) & \approx_{\mathrm{R}} & (6.5) \\
(\nu x)(\llbracket P \rrbracket\, a \mid \llbracket x{:=}Q \rrbracket) & \triangleq & \llbracket P\,\langle x{:=}Q \rangle \rrbracket\, a
\end{array}
$$

This shows that β-reduction is implemented in π by at least one synchronisation. Notice that, in step $(*)$, the process $(\nu c)\,(\llbracket c{:=}Q{\cdot}a \rrbracket) \triangleq (\nu c)\,(!\,c(v,d).(\llbracket v{:=}Q \rrbracket \mid !\,d{\rightarrow}a))$ is weakly bisimilar to 0.

This example stresses that all synchronisations in the image of $\llbracket \cdot \rrbracket\, \cdot$ are over hidden channels, so by Proposition 2.5 are in $\approx$.

Remark 6.8 We could not have represented the extensional rules: note that

$$\llbracket \lambda x.yx \rrbracket\, a \;\triangleq\; (\nu xb)((\nu c)\,(\llbracket y \rrbracket\, c \mid \llbracket c := x{\cdot}b \rrbracket) \mid \overline{a}\langle x,b \rangle)$$

is not weakly bisimilar to $\llbracket y \rrbracket\, a$, and neither is

$$\llbracket \mu\alpha.[\beta \rrbracket\, y]a \;\triangleq\; \llbracket y \rrbracket\, \beta[a/\alpha] \;=\; \llbracket y \rrbracket\, \beta$$

weakly bisimilar to:

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

We can also show that typeability is preserved: first, we need a substitution lemma.

Proposition 6.9 (Substitution [7, 8]) *If* $\mathsf{P} : \Gamma, x{:}A \vdash^{io}_{\pi} x{:}A, \Delta$, *then* $\mathsf{P}[b/x] : \Gamma, b{:}A \vdash^{io}_{\pi} b{:}A, \Delta$ *for* b *fresh or* $b{:}A \in \Gamma \cup \Delta$.

Proof: Easy. □

Theorem 6.10 ($\llbracket\cdot\rrbracket\cdot$ preserves $\lambda\mu\mathbf{x}$ types) *If* $\Gamma \vdash_{\mu\mathbf{x}} M : A \mid \Delta$, *then* $\llbracket M\rrbracket\, a : \Gamma \vdash^{io}_{\pi} a{:}A, \Delta$.

Proof: By induction on the structure of derivations in $\vdash_{\mu\mathbf{x}}$:

$((Ax))$: Then $M = x$, and $\Gamma = \Gamma', x{:}A$. Notice that $\llbracket x\rrbracket^{\mathrm{H}}\, a = x(u).!\,u{\twoheadrightarrow}a$, and that

$$\dfrac{\dfrac{\dfrac{}{u{\twoheadrightarrow}a : \Gamma', u{:}A \vdash^{io}_{\pi} a{:}A}\,(\twoheadrightarrow)}{!\,u{\twoheadrightarrow}a : \Gamma', u{:}A \vdash^{io}_{\pi} a{:}A}\,(!)}{x(u).!\,u{\twoheadrightarrow}a : \Gamma', x{:}A \vdash^{io}_{\pi} a{:}A}\,(in)$$

$((\to I))$: Then $M = \lambda x.N$, $A = C{\to}D$, and $\Gamma, x{:}C \vdash_{\mu\mathbf{x}} N : D \mid \Delta$; by definition, $\llbracket \lambda x.N\rrbracket\, a = (\nu xb)(\llbracket N\rrbracket\, b \mid \overline{a}\langle x,b\rangle)$. Then, by induction, $\mathcal{D} :: \llbracket N\rrbracket\, b : \Gamma, x{:}C \vdash^{io}_{\pi} b{:}D, \Delta$ exists, and we can construct:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \llbracket N\rrbracket\, b : \Gamma, x{:}C \vdash^{io}_{\pi} b{:}D, \Delta\end{array} \qquad \dfrac{}{\overline{a}\langle x,b\rangle : x{:}C \vdash^{io}_{\pi} a{:}C{\to}D, b{:}D}\,(\langle\rangle\text{-}out')}{\llbracket N\rrbracket\, b \mid \overline{a}\langle x,b\rangle : x{:}C \vdash^{io}_{\pi} a{:}C{\to}D, b{:}D, \Delta}\,(|)}{(\nu b)\,(\llbracket N\rrbracket\, b \mid \overline{a}\langle x,b\rangle) : \Gamma, x{:}C \vdash^{io}_{\pi} a{:}C{\to}D, \Delta}\,(\nu)}{(\nu xb)(\llbracket N\rrbracket\, b \mid \overline{a}\langle x,b\rangle) : \Gamma \vdash^{io}_{\pi} a{:}C{\to}D, \Delta}\,(\nu)$$

Notice that $(\nu xb)(\llbracket N\rrbracket\, b \mid \overline{a}\langle x,b\rangle) = \llbracket \lambda x.N\rrbracket\, a$.

$((\mu))$: Then $M = \mu\alpha.[\beta]N$, and either:

- $(\alpha \neq \beta)$: Then $\Gamma \vdash_{\mu\mathbf{x}} N : A \mid \alpha{:}A, \beta{:}B, \Delta$. By induction, there exist a derivation for $\llbracket N\rrbracket\, \beta : \Gamma \vdash^{io}_{\pi} \alpha{:}A, \beta{:}B, \Delta$; then by Lemma 6.9, also $\llbracket N\rrbracket\, \beta[a/\alpha] : \Gamma \vdash^{io}_{\pi} a{:}A, \beta{:}B, \Delta$ as well, and $\llbracket N\rrbracket\, \beta[a/\alpha] = \llbracket \mu\alpha.[\beta]N\rrbracket\, a$.
- $(\alpha = \beta)$: Then $\Gamma \vdash_{\mu\mathbf{x}} N : A \mid \alpha{:}A, \Delta$. By induction, there exist a derivation for $\llbracket N\rrbracket\, a : \Gamma \vdash^{io}_{\pi} a{:}A, \alpha{:}A, \Delta$; then by Lemma 6.9, also $\llbracket N\rrbracket\, \alpha[a/\alpha] : \Gamma \vdash^{io}_{\pi} a{:}A, \Delta$ as well, and $\llbracket N\rrbracket\, \alpha[a/\alpha] = \llbracket \mu\alpha.[\alpha]N\rrbracket\, a$.

$((C\text{-}cut))$: Then $M = P\,\langle \alpha := Q{\cdot}\gamma\rangle$ and we have $\Gamma \vdash_{\mu\mathbf{x}} P : C \mid \alpha{:}B{\to}A, \Delta$ and $\Gamma \vdash_{\mu\mathbf{x}} Q : A \mid \gamma{:}B, \Delta$ for some B. By induction, there exist derivations $\mathcal{D}_1 :: \llbracket P\rrbracket\, a : \Gamma \vdash^{io}_{\pi} a{:}C, \alpha{:}B{\to}A, \Delta$ and $\mathcal{D}_2 :: \llbracket Q\rrbracket\, w : \Gamma \vdash^{io}_{\pi} w{:}B, \Delta$, and we can construct the derivation

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_1\\ \llbracket P\rrbracket\, a : \Gamma \vdash^{io}_{\pi} a{:}B{\to}A, \Delta\end{array} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_2\\ \llbracket Q\rrbracket\, w : \Gamma \vdash^{io}_{\pi} w{:}B, \Delta\end{array}}{\overline{b}(w).\llbracket Q\rrbracket\, w : \Gamma \vdash^{io}_{\pi} b{:}B, w{:}B, \Delta}\,(\overline{\nu})}{!\,\overline{b}(w).\llbracket Q\rrbracket\, w : \Gamma \vdash^{io}_{\pi} b{:}B, \Delta}\,(!) \qquad \dfrac{\dfrac{}{d{\twoheadrightarrow}\gamma : d{:}A \vdash^{io}_{\pi} \gamma{:}A}\,(\twoheadrightarrow)}{!\,d{\twoheadrightarrow}\gamma : d{:}A \vdash^{io}_{\pi} \gamma{:}A}\,(!)}{\llbracket b := Q\rrbracket \mid !\,d{\twoheadrightarrow}\gamma : \Gamma, d{:}A \vdash^{io}_{\pi} \gamma{:}A, b{:}B, \Delta}\,(|)}{\alpha(b,d).(\llbracket b := Q\rrbracket \mid !\,d{\twoheadrightarrow}\gamma) : \Gamma, \alpha{:}B{\to}A \vdash^{io}_{\pi} \gamma{:}A, \Delta}\,(\langle\rangle\text{-}in)}{!\,\alpha(b,d).(\llbracket b := Q\rrbracket \mid !\,d{\twoheadrightarrow}\gamma) : \Gamma, \alpha{:}B{\to}A \vdash^{io}_{\pi} \gamma{:}A, \Delta}\,(!)}{\llbracket P\rrbracket\, a \mid !\,\alpha(b,d).(\llbracket b := Q\rrbracket \mid !\,d{\twoheadrightarrow}\gamma) : \Gamma, \alpha{:}B{\to}A \vdash^{io}_{\pi} \alpha{:}B{\to}A, \gamma{:}A, \Delta}\,(|)}{(\nu\alpha)(\llbracket P\rrbracket\, a \mid !\,\alpha(v,d).(\llbracket v := Q\rrbracket \mid !\,d{\twoheadrightarrow}\gamma)) : \Gamma \vdash^{io}_{\pi} \gamma{:}A, \Delta}\,(\nu)$$

and $(\nu\alpha)(\llbracket P \rrbracket\, a \mid !\alpha(v,d).(\llbracket v := Q \rrbracket \mid !d \rightarrow \gamma)) = \llbracket P\langle \alpha := Q\cdot\gamma\rangle \rrbracket\, a$.

$((\rightarrow E), (T\text{-}cut))$: Since $\llbracket PQ \rrbracket\, a = (\nu c)\,(\llbracket P \rrbracket\, c \mid \llbracket c := Q\cdot a \rrbracket)$ and $\llbracket M\,\langle x := N\rangle \rrbracket\, a = (\nu x)(\llbracket M \rrbracket\, a \mid \llbracket x := N \rrbracket)$, these cases are very similar to that for $(C\text{-}cut)$. □

7 Soundness, completeness, and termination

As in [37, 44], we can now show a reduction-preservation result for our encoding with respect to explicit head-reduction for $\lambda\mu\mathbf{x}$, by showing that $\llbracket\cdot\rrbracket\,\cdot$ preserves $\rightarrow_{\mathbf{xH}}$ up to weak bisimilarity. 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 $M \rightarrow_{\mathbf{xH}} N$, then $\llbracket M \rrbracket\, a \approx\!\!\rightarrow^*_{\pi} \llbracket N \rrbracket\, a$.*

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

(*Main reduction rules*):

$((\lambda x.M)N \rightarrow M\,\langle x := N\rangle)$: $\llbracket(\lambda x.M)N \rrbracket\, a \;\triangleq\; (\nu c)\,(\llbracket \lambda x.M \rrbracket\, c \mid \llbracket c := N\cdot a \rrbracket) \;\triangleq$
$(\nu c)\,((\nu xb)(\llbracket M \rrbracket\, b \mid \overline{c}\langle x,b\rangle) \mid !c(v,d).(\llbracket v := N \rrbracket \mid !d \rightarrow a)) \;\rightarrow_{\pi}\; (c)$
$(\nu cxb)\,(\llbracket M \rrbracket\, b \mid \llbracket x := N \rrbracket \mid \llbracket c := N\cdot a \rrbracket \mid !b \rightarrow a) \;\approx_{\mathrm{G}}$
$(\nu xb)\,(\llbracket M \rrbracket\, b \mid \llbracket x := N \rrbracket \mid !b \rightarrow a) \;\approx_{\mathrm{R}}(6.5)\; (\nu x)(\llbracket M \rrbracket\, a \mid \llbracket x := N \rrbracket) \;\triangleq$
$\llbracket M\langle x := N\rangle \rrbracket\, a$

$((\mu\beta.[\alpha]M)N \rightarrow \mu\gamma.([\alpha]M)\,\langle \beta := N\cdot\gamma\rangle,\ \gamma\ \textit{fresh})$: $\llbracket(\mu\beta.[\alpha]M)N \rrbracket\, a \;\triangleq$
$(\nu c)\,(\llbracket \mu\beta.[\alpha]M \rrbracket\, c \mid \llbracket c := N\cdot a \rrbracket) \;\triangleq$
$(\nu c)\,(\llbracket M \rrbracket\, \alpha[c/\beta] \mid \llbracket c := N\cdot a \rrbracket) \;=_{\alpha}\; (c\ \textit{fresh})$
$(\nu\beta)(\llbracket M \rrbracket\, \alpha \mid \llbracket \beta := N\cdot a \rrbracket) \;=$
$(\nu\beta)(\llbracket M \rrbracket\, \alpha \mid \llbracket \beta := N\cdot\gamma \rrbracket)\,[a/\gamma] \;\equiv\; (s\ \textit{fresh})$
$(\nu s)\,(\nu\beta)(\llbracket M \rrbracket\, \alpha \mid \llbracket \beta := N\cdot\gamma \rrbracket)\,[a/\gamma] \;\triangleq\; (\nu s)\,(\nu\beta)(\llbracket [\alpha]M \rrbracket\, s \mid \llbracket \beta := N\cdot\gamma \rrbracket)\,[a/\gamma] \;\triangleq$
$(\nu s)\,\llbracket([\alpha]M)\,\langle \beta := N\cdot\gamma\rangle \rrbracket\, s\,[a/\gamma] \;\triangleq\; \llbracket \mu\gamma.([\alpha]M)\,\langle \beta := N\cdot\gamma\rangle \rrbracket\, a$

$(\mu\beta.[\beta]M \rightarrow M,\ \textit{if}\ \beta \notin fn(M))$: $\llbracket \mu\beta.[\beta]M \rrbracket\, a \;\triangleq\; \llbracket M \rrbracket\, \beta[a/\beta] \;=\; (\beta \notin fn(M))\ \llbracket M \rrbracket\, a$

$(\mu\alpha.[\beta]\mu\gamma.[\delta]M \rightarrow \mu\alpha.[\delta]M[\beta/\gamma], \gamma \neq \delta)$: $\llbracket \mu\alpha.[\beta]\mu\gamma.[\delta]M \rrbracket\, a \;\triangleq\; \llbracket \mu\gamma.[\delta]M \rrbracket\, \beta[a/\alpha] \;\triangleq$
$\llbracket M \rrbracket\, \delta[\beta/\gamma][a/\alpha] \;=\; \llbracket M[\beta/\gamma] \rrbracket\, \delta[a/\alpha] \;\triangleq\; \llbracket \mu\alpha.[\delta]M[\beta/\gamma] \rrbracket\, a$

$(\mu\alpha.[\beta]\mu\gamma.[\gamma]M \rightarrow \mu\alpha.[\beta]M[\beta/\gamma])$: $\llbracket \mu\alpha.[\beta]\mu\gamma.[\gamma]M \rrbracket\, a \;\triangleq\; \llbracket \mu\gamma.[\gamma]M \rrbracket\, \beta[a/\alpha] \;\triangleq$
$\llbracket M \rrbracket\, \gamma[\beta/\gamma][a/\alpha] \;=\; \llbracket M[\beta/\gamma] \rrbracket\, \beta[a/\alpha] \;\triangleq\; \llbracket \mu\alpha.[\beta]M[\beta/\gamma] \rrbracket\, a$

(*Term substitution rules*):

$(x\,\langle x := N\rangle \rightarrow N)$: $\llbracket x\langle x := N\rangle \rrbracket\, a \;\triangleq\; (\nu x)(\llbracket x \rrbracket\, a \mid \llbracket x := N \rrbracket) \;\triangleq$
$(\nu x)(x(u).!u \rightarrow a \mid !\overline{x}(w).\llbracket N \rrbracket\, w) \;\rightarrow_{\pi}(x)\; (\nu w)\,(!w \rightarrow a \mid \llbracket N \rrbracket\, w) \mid (\nu x)\,(\llbracket x := N \rrbracket) \;\approx_{\mathrm{G}}$
$(\nu w)\,(!w \rightarrow a \mid \llbracket N \rrbracket\, w) \;\approx_{\mathrm{R}}(6.5)\; \llbracket N \rrbracket\, a$

$(M\,\langle x := N\rangle \rightarrow M,\ x \notin fv(M))$: $\llbracket M\langle x := N\rangle \rrbracket\, a \;\triangleq\; (\nu x)(\llbracket M \rrbracket\, a \mid \llbracket x := N \rrbracket) \;\triangleq$
$(\nu x)(\llbracket M \rrbracket\, a \mid !\overline{x}(w).\llbracket N \rrbracket\, w) \;\equiv (a \neq x)\; \llbracket M \rrbracket\, a \mid (\nu x)\,(!\overline{x}(w).\llbracket N \rrbracket\, w) \;\approx_{\mathrm{G}}\; \llbracket M \rrbracket\, a$

$((\lambda y.M)\,\langle x := N\rangle \rightarrow \lambda y.(M\,\langle x := N\rangle),\ x = hv(M))$: $\llbracket(\lambda y.M)\,\langle x := N\rangle \rrbracket\, a \;\triangleq$
$(\nu x)((\nu yb)(\llbracket M \rrbracket\, b \mid \overline{a}\langle y,b\rangle) \mid \llbracket x := N \rrbracket) \;\equiv (a \neq x)$
$(\nu yb)((\nu x)(\llbracket M \rrbracket\, b \mid \llbracket x := N \rrbracket) \mid \overline{a}\langle y,b\rangle) \;\triangleq\; \llbracket \lambda y.M\,\langle x := N\rangle \rrbracket\, a$

$((PQ)\,\langle x := N\rangle \rightarrow (P\,\langle x := N\rangle Q)\,\langle x := N\rangle,\ x = hv(PQ))$: $\llbracket(PQ)\,\langle x := N\rangle \rrbracket\, a \;\triangleq$

$$
\begin{array}{ll}
(\nu x)((\nu c)(\llbracket P \rrbracket c \mid \llbracket c := Q\cdot a \rrbracket) \mid !\overline{x}(w).\llbracket N \rrbracket w) & \approx \ (2.6) \\
(\nu x)((\nu c)((\nu x)(\llbracket P \rrbracket c \mid !\overline{x}(w).\llbracket N \rrbracket w) \mid \llbracket c := Q\cdot a \rrbracket) \mid !\overline{x}(w).\llbracket N \rrbracket w) & \stackrel{\Delta}{=} \\
(\nu x)((\nu c)(\llbracket P\langle x := N\rangle \rrbracket c \mid \llbracket c := Q\cdot a \rrbracket) \mid \llbracket x := N \rrbracket) & \stackrel{\Delta}{=} \\
\llbracket (P\langle x := N\rangle Q)\langle x := N\rangle \rrbracket a &
\end{array}
$$

$((\mu\alpha.[\beta]M)\langle x := N\rangle \to \mu\alpha.[\beta](M\langle x := N)\rangle,\ x = hv(M))$:

$$
\begin{array}{lll}
\llbracket (\mu\alpha.[\beta]M)\langle x := N\rangle \rrbracket a & \stackrel{\Delta}{=} & \\
(\nu x)(\llbracket M \rrbracket \beta[a/\alpha] \mid \llbracket x := N \rrbracket) & \equiv (\alpha \notin fn(N)) & (\nu x)(\llbracket M \rrbracket \beta \mid \llbracket x := N \rrbracket)[a/\alpha] \stackrel{\Delta}{=} \\
\llbracket M\langle x := N\rangle \rrbracket \beta[a/\alpha] & \stackrel{\Delta}{=} & \llbracket \mu\alpha.[\beta]M\langle x := N\rangle \rrbracket a
\end{array}
$$

(*Structural rules*):

$((\mu\delta.[\alpha]M)\langle \alpha := N\cdot\gamma\rangle \to \mu\delta.[\gamma](M\langle \alpha := N\cdot\gamma\rangle)N)$:

$$
\begin{array}{lll}
 & & \llbracket (\mu\delta.[\alpha]M)\langle \alpha := N\cdot\gamma\rangle \rrbracket a \stackrel{\Delta}{=} \\
(\nu\alpha)(\llbracket M \rrbracket \alpha[a/\delta] \mid \llbracket \alpha := N\cdot\gamma \rrbracket) & = & (\delta \notin fn(\langle \alpha := N\cdot\gamma\rangle)) \\
(\nu\alpha)(\llbracket M \rrbracket \alpha \mid \llbracket \alpha := N\cdot\gamma \rrbracket)[a/\delta] & \approx, =_\alpha & \\
(\nu c)((\nu\alpha)(\llbracket M \rrbracket c \mid \llbracket \alpha := N\cdot\gamma \rrbracket) \mid \llbracket c := N\cdot\gamma \rrbracket)[a/\delta] & \stackrel{\Delta}{=} & \\
(\nu c)(\llbracket M\langle \alpha := N\cdot\gamma\rangle \rrbracket c \mid \llbracket c := N\cdot\gamma \rrbracket)[a/\delta] & \stackrel{\Delta}{=} & \llbracket \mu\delta.[\gamma](M\langle \alpha := N\cdot\gamma\rangle)N \rrbracket a
\end{array}
$$

$(M\langle \alpha := N\cdot\gamma\rangle \to M,\ \alpha \notin fn(M))$: $\llbracket M\langle \alpha := N\cdot\gamma\rangle \rrbracket a \stackrel{\Delta}{=} (\nu\alpha)(\llbracket M \rrbracket a \mid \llbracket \alpha := N\cdot\gamma \rrbracket) \equiv (\alpha \neq a)$

$$
\begin{array}{ll}
\llbracket M \rrbracket a \mid (\nu\alpha)(\llbracket \alpha := N\cdot\gamma \rrbracket) & \approx_{\mathrm{G}} \\
\llbracket M \rrbracket a &
\end{array}
$$

(*Substitution rule*):

$(M\langle x := N\rangle\langle y := P\rangle \to M\langle y := P\rangle\langle x := N\rangle\langle y := P\rangle)$: $\llbracket M\langle x := N\rangle\langle y := P\rangle \rrbracket a \stackrel{\Delta}{=}$

$$
\begin{array}{ll}
(\nu y)((\nu x)(\llbracket M \rrbracket a \mid \llbracket x := N \rrbracket) \mid !\overline{y}(w).\llbracket P \rrbracket w) & \approx \ (2.6) \\
(\nu y)((\nu x)((\nu y)(\llbracket M \rrbracket a \mid !\overline{y}(w).\llbracket P \rrbracket w) \mid \llbracket x := N \rrbracket) \mid !\overline{y}(w).\llbracket P \rrbracket w) & \stackrel{\Delta}{=} \\
\llbracket M\langle y := P\rangle\langle x := N\rangle\langle y := P\rangle \rrbracket a &
\end{array}
$$

(*Contextual rules*): By induction. □

Remark that, by Property 2.5, all proper reductions in this proof are in $\approx$. Also, the proof shows that β-reduction is implemented in π by at least one synchronisation.

Example 7.2 Notice that we could have demanded that the contextual rules of Definition 5.2(*iv*) be stated as:

$$
M \to N \ \Rightarrow \ \begin{cases}
\lambda x.M & \to \ \lambda x.N & \\
\mu\alpha.[\beta]M & \to \ \mu\alpha.[\beta]N & (\alpha \neq \beta,\ M \neq \mu\gamma.[\delta]M') \\
ML & \to \ NL & (ML \textit{ not a redex}) \\
M\langle x := L\rangle & \to \ N\langle x := L\rangle & (x \neq hv(M)) \\
M\langle \alpha := L\cdot\gamma\rangle & \to \ N\langle \alpha := L\cdot\gamma\rangle & (\alpha \neq hn(M))
\end{cases}
$$

which would imply that we would only allow the reduction

$$
(\mu\alpha.[\alpha](\lambda z.M)N)L \ \to \ \mu\gamma.[\gamma]((\lambda z.M)N)L
$$

and not

$$
(\mu\alpha.[\alpha](\lambda z.M)N)L \ \to \ (\mu\alpha.[\alpha]M\langle z := N\rangle)L
$$

However, since under the interpretation μ-abstractions and naming are modelled using congruence and syntactic operations, we have the following:

$$
\begin{array}{ll}
\llbracket (\mu\alpha.[\alpha](\lambda z.M)N)L \rrbracket a & \stackrel{\Delta}{=} \\
(\nu\alpha)((\nu e)(\llbracket \lambda z.M \rrbracket e \mid \llbracket e := N\cdot\alpha \rrbracket) \mid \llbracket \alpha := L\cdot a \rrbracket) & \stackrel{\Delta}{=} \\
(\nu\alpha)((\nu e)((\nu zb)(\llbracket M \rrbracket b \mid \overline{e}\langle z,b\rangle) \mid !e(v,d).(\llbracket v := N \rrbracket \mid !d\to\alpha)) \mid !\alpha(v,d).(\llbracket v := L \rrbracket \mid !d\to a)) &
\end{array}
$$

Notice that synchronisation over e is possible, so the contraction of the redex $(\lambda z.P)N$ is modelled under the interpretation, which justifies that we allow this redex to be contracted under $\to_{\mathrm{XH}}$, although it is encompassed by the redex $(\mu\alpha.[\alpha](\lambda z.M)N)L$.

Similarly, we have

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

and in the process

$$\begin{array}{l} \llbracket(\lambda x.((\lambda z.M)N))L\rrbracket\, a \quad \triangleq \\ (\nu c)\,((\nu xb)((\nu e)\,((\nu zb)(\llbracket M\rrbracket\, b \mid \overline{e}\langle z{,}b\rangle) \mid \\ \qquad !e(v{,}d).(\llbracket v{:=}N\rrbracket \mid !d{\to}b)) \mid \overline{c}\langle x{,}b\rangle) \mid !c(v{,}d).(\llbracket v{:=}L\rrbracket \mid !d{\to}a)) \end{array}$$

both synchronisations over c and e are possible.

We can now easily show:

Theorem 7.3 (Operational Soundness for $\to_{\mathrm{xH}}$) *i)* $M \to^*_{\mathrm{xH}} N \Rightarrow \llbracket M\rrbracket\, a \approx \llbracket N\rrbracket\, a$.
ii) If $M\uparrow_{\mathrm{xH}}$ *then* $\llbracket M\rrbracket\, a\uparrow$.

Proof: The first is shown by induction using Theorem 7.1, using Proposition 2.5; the second follows from Example 6.7, and the fact that μ-reduction and substitution do not loop [42] (*i.e.* non-termination is caused only by β-reduction). □

We can also show:

Theorem 7.4 (Operational Completeness for $\to_{\mathrm{xH}}$) *i) If* $\llbracket M\rrbracket\, a \to_\pi \mathsf{P}$, *then there exists* N *such that* $\mathsf{P} \approx \llbracket N\rrbracket\, a$ *and* $M \to^*_{\mathrm{xH}} N$.
ii) If $\llbracket M\rrbracket\, a \to^*_\pi \llbracket N\rrbracket\, a$ *then* $M \to^*_{\mathrm{xH}} N$.

Proof: The fist is shown by easy induction on the structure of terms, using the fact that all synchronisations that are possible in $\llbracket M\rrbracket\, a$ are generated by the interpretation, and correspond to reductions in $\to_{\mathrm{xH}}$. The second follows from Lemma 1.8 and 4.2, and the first part. □

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

Theorem 7.5 *If* $M =_{\mathrm{x}} N$, *then* $\llbracket M\rrbracket\, a \approx \llbracket N\rrbracket\, a$.

Proof: By induction on the definition of $=_{\mathrm{x}}$; we only show the cases that are different from the proof of Theorem 7.1.

$$\begin{array}{lll} ((PQ)\langle x{:=}N\rangle \to (P\langle x{:=}N\rangle)(Q\langle x{:=}N\rangle)): & \llbracket(PQ)\langle x{:=}N\rangle\rrbracket\, a & \triangleq \\ \quad (\nu x)((\nu c)\,(\llbracket P\rrbracket\, c \mid !c(v{,}d).(\llbracket v{:=}Q\rrbracket \mid !d{\to}a)) \mid !\overline{x}(w).\llbracket N\rrbracket\, w) & & \approx \quad (2.6) \\ \quad (\nu c)\,((\nu x)\,(\llbracket P\rrbracket\, c \mid \llbracket x{:=}N\rrbracket) \mid (\nu x)\,(!c(v{,}d).(!\overline{v}(w).\llbracket Q\rrbracket\, w \mid d{\to}a) \mid \llbracket x{:=}N\rrbracket)) & & \approx \quad (2.6) \\ \quad (\nu c)\,((\nu x)\,(\llbracket P\rrbracket\, c \mid \llbracket x{:=}N\rrbracket) \mid !c(v{,}d).((\nu x)\,(!\overline{v}(w).\llbracket Q\rrbracket\, w \mid \llbracket x{:=}N\rrbracket) \mid d{\to}a)) & & \approx \quad (2.6) \\ \quad (\nu c)\,((\nu x)\,(\llbracket P\rrbracket\, c \mid \llbracket x{:=}N\rrbracket) \mid !c(v{,}d).(!\overline{v}(w).(\nu x)(\llbracket Q\rrbracket\, w \mid \llbracket x{:=}N\rrbracket) \mid d{\to}a)) & & \triangleq \\ \quad (\nu c)\,(\llbracket P\langle x{:=}N\rangle\rrbracket\, c \mid !c(v{,}d).(!\overline{v}(w).\llbracket Q\langle x{:=}N\rangle\rrbracket\, w \mid d{\to}a)) & & \triangleq \\ \quad (\nu c)\,(\llbracket P\langle x{:=}N\rangle\rrbracket\, c \mid !c(v{,}d).(\llbracket v{:=}Q\langle x{:=}N\rangle\rrbracket \mid d{\to}a)) & & \triangleq \\ \quad (\nu c)\,(\llbracket P\langle x{:=}N\rangle\rrbracket\, c \mid \llbracket c := Q\langle x{:=}N\rangle{\cdot}a\rrbracket) & & \triangleq \\ \quad \llbracket(P\langle x{:=}N\rangle)(Q\langle x{:=}N\rangle)\rrbracket\, a & & \end{array}$$

$$\begin{array}{lll} ((\mu\delta.[\beta]M)\langle \alpha{:=}N{\cdot}\gamma\rangle \to \mu\delta.[\beta](M\langle \alpha{:=}N{\cdot}\gamma\rangle)\ \alpha \neq \beta): & \llbracket(\mu\delta.[\beta]M)\langle \alpha{:=}N{\cdot}\gamma\rangle\rrbracket\, a & \triangleq \\ \quad (\nu\alpha)(\llbracket M\rrbracket\, \beta[a/\delta] \mid \llbracket \alpha{:=}N{\cdot}\gamma\rrbracket) & = \ (\delta \notin fn(\langle \alpha{:=}N{\cdot}\gamma\rangle)) & \\ \quad (\nu\alpha)(\llbracket M\rrbracket\, \beta \mid \llbracket \alpha{:=}N{\cdot}\gamma\rrbracket)\,[a/\delta] & \triangleq \ \llbracket \mu\delta.[\beta](M\langle \alpha{:=}N{\cdot}\gamma\rangle)\rrbracket\, a & \end{array}$$

$$
\begin{array}{lll}
((\lambda x.M)\langle\alpha:=N\cdot\gamma\rangle \to \lambda x.M\langle\alpha:=N\cdot\gamma\rangle):\llbracket(\lambda x.M)\langle\alpha:=N\cdot\gamma\rangle\rrbracket\, a & \triangleq & \\
\quad (\nu\alpha)((\nu xb)(\llbracket M\rrbracket\, b \mid \overline{a}\langle x,b\rangle) \mid \llbracket\alpha:=N\cdot\gamma\rrbracket) & \equiv & (\alpha \neq x,b) \\
\quad (\nu xb)((\nu\alpha)(\llbracket M\rrbracket\, b \mid \llbracket\alpha:=N\cdot\gamma\rrbracket) \mid \overline{a}\langle x,b\rangle) & \triangleq & \llbracket\lambda x.M\langle\alpha:=N\cdot\gamma\rangle\rrbracket\, a
\end{array}
$$

$$
\begin{array}{lll}
((PQ)\langle\alpha:=N\cdot\gamma\rangle \to (P\langle\alpha:=N\cdot\gamma\rangle)(Q\langle\alpha:=N\cdot\gamma\rangle)): & \llbracket(PQ)\langle\alpha:=N\cdot\gamma\rangle\rrbracket\, a & \triangleq \\
\quad (\nu\alpha)((\nu c)(\llbracket P\rrbracket\, c \mid !c(v,d).(!\overline{v}(w).\llbracket Q\rrbracket\, w \mid !d\to a)) \mid !\alpha(v,d).(!\overline{v}(w).\llbracket N\rrbracket\, w \mid !d\to\gamma)) & \approx & (2.6) \\
\quad (\nu c)((\nu\alpha)(\llbracket P\rrbracket\, c \mid \llbracket\alpha:=N\cdot\gamma\rrbracket) \mid (\nu\alpha)(!c(v,d).(!\overline{v}(w).\llbracket Q\rrbracket\, w \mid !d\to a) \mid \llbracket\alpha:=N\cdot\gamma\rrbracket)) & \approx & (2.6) \\
\quad (\nu c)((\nu\alpha)(\llbracket P\rrbracket\, c \mid \llbracket\alpha:=N\cdot\gamma\rrbracket) \mid !c(v,d).((\nu\alpha)(!\overline{v}(w).\llbracket Q\rrbracket\, w \mid \llbracket\alpha:=N\cdot\gamma\rrbracket \mid !d\to a))) & \approx & (2.6) \\
\quad (\nu c)((\nu\alpha)(\llbracket P\rrbracket\, c \mid \llbracket\alpha:=N\cdot\gamma\rrbracket) \mid !c(v,d).(!\overline{v}(w).(\nu\alpha)(\llbracket Q\rrbracket\, w \mid \llbracket\alpha:=N\cdot\gamma\rrbracket) \mid !d\to a)) & \triangleq & \\
\quad (\nu c)(\llbracket P\langle\alpha:=N\cdot\gamma\rangle\rrbracket\, c \mid !c(v,d).(!\overline{v}(w).\llbracket Q\langle\alpha:=N\cdot\gamma\rangle\rrbracket\, w \mid !d\to a)) & \triangleq & \\
\quad (\nu c)(\llbracket P\langle\alpha:=N\cdot\gamma\rangle\rrbracket\, c \mid !c(v,d).(\llbracket v:=Q\langle\alpha:=N\cdot\gamma\rangle\rrbracket \mid !d\to a)) & \triangleq & \\
\quad \llbracket(P\langle\alpha:=N\cdot\gamma\rangle)(Q\langle\alpha:=N\cdot\gamma\rangle)\rrbracket\, a & &
\end{array}
$$

The steps to a reflexive, transitive, contextual closure and equivalence relation follow directly from the fact that '$\approx$' is a congruence. □

Now the following is an immediate consequence:

Theorem 7.6 (Semantics) *If $M =_{\beta\mu} N$, then $\llbracket M\rrbracket^{\text{H}}\, a \approx \llbracket N\rrbracket^{\text{H}}\, a$.*

Proof: By induction on the definition of $=_{\beta\mu}$. The case $M \to^*_{\beta\mu} N$ follows from the fact that then, by Proposition 4.2, also $M \to^*_{\mathbf{x}} N$, so by Theorem 7.5, we have $\llbracket M\rrbracket^{\text{H}}\, a \approx \llbracket N\rrbracket^{\text{H}}\, a$. 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 0 (see also Lemma 8.8), but are not all related through $=_{\beta\mu}$. However, using a notion of weak equivalence, we can deal with the reverse part and will do so in the last sections of this paper.

We can show that interpretation of terms in $\to_{\mathbf{x}\text{H}}$-normal form are in normal form as well, which is a property that we need here.

Lemma 7.7 If $\mathbf{N}$ is a $\to_{\mathbf{x}\text{H}}$-normal form, then $\llbracket\mathbf{N}\rrbracket\, a$ is irreducible.

Proof: By induction on the structure of terms in $\to_{\mathbf{x}\text{H}}$-normal form.

$$
\begin{array}{lll}
(\mathbf{N} = xM_1\cdots M_n\ (n \geq 0)): \ \text{Then } \llbracket xM_1\cdots M_n\rrbracket\, a & \triangleq & \\
\quad (\nu c_n)(\llbracket xM_1\cdots M_{n-1}\rrbracket\, c_n \mid \llbracket c_n:=M_n\cdot a\rrbracket) & \triangleq & \\
\quad (\nu c_n)((\nu c_{n-1})(\llbracket xM_1\cdots M_{n-2}\rrbracket\, c_{n-1} \mid \llbracket c_{n-1}:=M_{n-1}\cdot c_n\rrbracket) \mid \llbracket c_n:=M_n\cdot a\rrbracket) & \equiv & \\
\quad (\nu c_n c_{n-1})(\llbracket xM_1\cdots M_{n-2}\rrbracket\, c_{n-1} \mid \llbracket c_{n-1}:=M_{n-1}\cdot c_n\rrbracket \mid \llbracket c_n:=M_n\cdot a\rrbracket) & \triangleq \ \cdots \ \triangleq & \\
\quad (\nu c_n\cdots c_1)(\llbracket x\rrbracket\, c_1 \mid \llbracket c_1:=M_1\cdot c_2\rrbracket \mid \cdots \mid \llbracket c_n:=M_n\cdot a\rrbracket) &&
\end{array}
$$

Since

$$
\begin{array}{rcl}
\llbracket x\rrbracket\, c_1 & = & x(u).!u\to c_1 \\
\llbracket c_i:=M_i\cdot c_{i+1}\rrbracket & = & !c_i(v,d).(!\overline{v}(w).\llbracket M_i\rrbracket\, w \mid !d\to c_{i+1})
\end{array}
$$

all $\llbracket M_i\rrbracket\, w$ appear under input on c_i, so no synchronisation inside one of the $\llbracket M_i\rrbracket\, w$ is possible; since all c_i are fresh, all are different from x and no synchronisation is possible over any of the c_i. So this process is in normal form.

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

$(\mathbf{N} = \mu\alpha.[\beta]\mathbf{N}'\ (\alpha \neq \beta \vee \alpha \in \mathbf{N}', \mathbf{N}' \neq \mu\gamma.[\delta]\mathbf{N}''))$: Then $\llbracket\mu\alpha.[\beta]\mathbf{N}'\rrbracket\, a \triangleq \llbracket\mathbf{N}'\rrbracket\, \beta[a/\alpha]$; this follows immediately by induction.

$(\mathbf{N} = \mathbf{N}'\langle x:=M\rangle\ (hv(\mathbf{N}') \neq x))$: Then $\llbracket\mathbf{N}'\langle x:=M\rangle\rrbracket\, a \triangleq (\nu x)(\llbracket\mathbf{N}'\rrbracket\, a \mid !\overline{x}(w).\llbracket M\rrbracket\, w)$. By induction, $\llbracket\mathbf{N}'\rrbracket\, a$ is in normal form; since x is not the head-variable of $\mathbf{N}$, the process $\llbracket\mathbf{N}'\rrbracket\, a$ has

no reachable input over x, so no synchronisation is possible over x; also, no synchronisation is possible inside $\llbracket M \rrbracket\, w$, as above.

$(\boldsymbol{N} = \boldsymbol{N}'\,\langle \alpha := M \cdot \gamma \rangle\ \ (hn\,(\boldsymbol{N}') \neq \alpha))$: Then $\llbracket \boldsymbol{N}'\,\langle \alpha := M \cdot \gamma \rangle \rrbracket\, a \triangleq (\nu\alpha)(\llbracket \boldsymbol{N}' \rrbracket\, a \mid !\,\alpha(v,d).(!\,\overline{v}(w).\llbracket M \rrbracket\, w \mid !\,d \rightarrow \gamma))$. By induction, $\llbracket \boldsymbol{N}' \rrbracket\, a$ is in normal form. Note that α is only a reachable output in $\llbracket \boldsymbol{N}' \rrbracket\, a$ if $\boldsymbol{N}'$ is an abstraction and $a = \alpha$; this is impossible, since a is fresh. As above, no synchronisation is possible inside $\llbracket M \rrbracket\, w$. □

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

We can now show the following termination results:

Theorem 7.8 (Termination) *i) If $M \rightarrow^{nf}_{\mathbf{xH}} N$, then $\llbracket M \rrbracket\, a \downarrow_\pi$.*

ii) If $M \rightarrow^{hnf}_{\beta\mu} N$, then $\llbracket M \rrbracket\, a \downarrow_\pi$.

Proof: *i*) By Lemma 7.7, if N is in explicit head-normal from, then $\llbracket N \rrbracket\, a$ is in normal form, and by Theorem 7.1, $\llbracket M \rrbracket\, a \rightarrow^*_\pi P$ with $P \approx \llbracket N \rrbracket\, a$. Since in the proof of Theorem 7.1, $\approx_{\mathrm{G}}$ only removes processes in normal form, this implies that P is in normal form.

ii) By Lemma 1.8, there exists L in HNF such that $M \rightarrow^{nf}_{\mathrm{H}} L$; by Property 5.4, there exists $\boldsymbol{N}$ such that $M \rightarrow^{nf}_{\mathbf{xH}} \boldsymbol{N}$; by the previous part, $\llbracket M \rrbracket\, a \downarrow_\pi$. □

Notice also that this result is stronger than the formulation of the termination result for Milner's interpretation in [44], 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 7.8 immediately leads to:

Corollary 7.9 *If $M \downarrow_{\beta\mu}$, then $\llbracket M \rrbracket\, a \downarrow_\pi$.*

8 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* [48, 49]). 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 is *weak* reduction.

Generally, the concept of *weak* reduction refers to the variant of calculi that eliminate the contextual rule

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

For the λ-calculus, then a closed normal form is an abstraction. Note that, in the context of $\lambda\mu$, this is no longer the case, since also context switches might occur inside the term.

$$
\begin{array}{lll}
{}^{\Gamma}\!\Delta\Delta\!{}_{\rfloor}\, a & \stackrel{\Delta}{=} & (\nu c)\,((\nu xb)({}^{\Gamma}\!xx\!{}_{\rfloor}\, b \mid \overline{c}\langle x,b\rangle) \mid !\,c(v,d).({}^{\Gamma}\!v := \lambda x.xx\!{}_{\rfloor} \mid !\,d \rightarrow a)) \qquad \rightarrow_{\pi} (c) \\
(\nu x)({}^{\Gamma}\!xx\!{}_{\rfloor}\, a \mid {}^{\Gamma}\!x := \lambda y.yy\!{}_{\rfloor}) & & \stackrel{\Delta}{=} \\
{}^{\Gamma}\!xx\,\langle x := \lambda y.yy\rangle\!{}_{\rfloor}\, a & & \stackrel{\Delta}{=} \\
(\nu x)((\nu c)\,(x(u).!\,u \rightarrow c \mid {}^{\Gamma}\!c := x\cdot a\!{}_{\rfloor}) \mid !\,\overline{x}(w).(\nu yb)({}^{\Gamma}\!yy\!{}_{\rfloor}\, b \mid \overline{w}\langle y,b\rangle)) & & \rightarrow_{\pi} (x,w) \\
(\nu x)((\nu c)\,((\nu yb)({}^{\Gamma}\!yy\!{}_{\rfloor}\, b \mid \overline{c}\langle y,b\rangle) \mid !\,c(v,d).({}^{\Gamma}\!v := x\!{}_{\rfloor} \mid !\,d \rightarrow a)) \mid {}^{\Gamma}\!x := \lambda y.yy\!{}_{\rfloor}) & & \stackrel{\Delta}{=} \\
{}^{\Gamma}\!(\lambda y.yy)x\,\langle x := \lambda y.yy\rangle\!{}_{\rfloor}\, a & & \rightarrow_{\pi} (c) \\
(\nu x)((\nu yb)({}^{\Gamma}\!yy\!{}_{\rfloor}\, b \mid {}^{\Gamma}\!y := x\!{}_{\rfloor} \mid !\,d \rightarrow a \mid (\nu c)(!\,c(v,d).({}^{\Gamma}\!v := x\!{}_{\rfloor} \mid !\,d \rightarrow a))) \mid {}^{\Gamma}\!x := \lambda y.yy\!{}_{\rfloor}) & & \approx \\
(\nu x)((\nu y)((\nu c)\,(y(u).!\,u \rightarrow c \mid {}^{\Gamma}\!c := y\cdot a\!{}_{\rfloor}) \mid !\,\overline{y}(w).{}^{\Gamma}\!x\!{}_{\rfloor}\, w) \mid {}^{\Gamma}\!x := \lambda y.yy\!{}_{\rfloor}) & & \stackrel{\Delta}{=} \\
{}^{\Gamma}\!yy\,\langle y := x\rangle\,\langle x := \lambda y.yy\rangle\!{}_{\rfloor}\, a & & \rightarrow_{\pi} (y) \\
(\nu x)((\nu cw)(!\,w \rightarrow c \mid {}^{\Gamma}\!c := y\cdot a\!{}_{\rfloor} \mid {}^{\Gamma}\!x\!{}_{\rfloor}\, w) \mid {}^{\Gamma}\!y := x\!{}_{\rfloor} \mid {}^{\Gamma}\!x := \lambda y.yy\!{}_{\rfloor}) & & \approx \\
(\nu x)((\nu y)((\nu c)\,({}^{\Gamma}\!x\!{}_{\rfloor}\, c \mid {}^{\Gamma}\!c := y\cdot a\!{}_{\rfloor}) \mid {}^{\Gamma}\!y := x\!{}_{\rfloor}) \mid {}^{\Gamma}\!x := \lambda y.yy\!{}_{\rfloor}) & & \stackrel{\Delta}{=} \\
{}^{\Gamma}\!xy\,\langle y := x\rangle\,\langle x := \lambda y.yy\rangle\!{}_{\rfloor}\, a & & \ldots
\end{array}
$$

Figure 1: Running ${}^{\Gamma}\!(\lambda x.xx)(\lambda x.xx)\!{}_{\rfloor}\, a$

Example 8.1 Consider the reduction of $\Delta\Delta$ that was given in Example 5.5; by Theorem 7.1, we have that ${}^{\Gamma}\!\Delta\Delta\!{}_{\rfloor}\, a \approx {}^{\Gamma}\!xy\,\langle y := x\rangle\,\langle x := \lambda y.yy\rangle\!{}_{\rfloor}\, a$ as shown in Figure 1, which shows that the interpretation of $\Delta\Delta$ reduces without creating output over a – it always occurs inside a subprocess of the shape

$$!\,c(v,d).({}^{\Gamma}\!v := y\!{}_{\rfloor} \mid !\,d \rightarrow a)$$

and does not input, since the head-variable is always bound, so ${}^{\Gamma}\!\Delta\Delta\!{}_{\rfloor}\, a$ is weakly bisimilar to $\mathsf{0}$ (see also Lemma 8.8). Therefore,

$$
\begin{array}{lll}
{}^{\Gamma}\!\lambda z.\Delta\Delta\!{}_{\rfloor}\, a \stackrel{\Delta}{=} & (\nu zb)({}^{\Gamma}\!\Delta\Delta\!{}_{\rfloor}\, b \mid \overline{a}\langle z,b\rangle) & \approx \\
 & (\nu zb)\,(\mathsf{0} \mid \overline{a}\langle z,b\rangle) & \approx \\
 & (\nu zb)({}^{\Gamma}\!\Omega\Omega\!{}_{\rfloor}\, b \mid \overline{a}\langle z,b\rangle) & \stackrel{\Delta}{=} \quad {}^{\Gamma}\!\lambda z.\Omega\Omega\!{}_{\rfloor}\, a
\end{array}
$$

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

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

Definition 8.2 We define the notion $\rightarrow_{w\beta\mu}$ of *weak $\beta\mu$-reduction* as in Definition 1.4, the notion $\rightarrow_{w\mathrm{H}}$ of *weak head reduction*[17] on $\lambda\mu$ as in Definition 1.7, and the notion $\rightarrow_{w\mathbf{x}\mathrm{H}}$ of *weak explicit head-reduction* on $\lambda\mu\mathbf{x}$ as in Definition 5.2, by (also) eliminating the rules:

$$
\begin{array}{rcl}
(\lambda y.M)\,\langle x := N\rangle & \rightarrow & \lambda y.(M\,\langle x := N\rangle) \\
(\lambda x.M)\,\langle \alpha := N\cdot\gamma\rangle & \rightarrow & \lambda x.(M\,\langle \alpha := N\cdot\gamma\rangle) \\
M \rightarrow N & \Rightarrow & \lambda x.M \rightarrow \lambda x.N
\end{array}
$$

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

Definition 8.3 (Weak head-normal forms for $\lambda\mu$) *i*) The $\lambda\mu$ *weak head-normal forms* (whnf) are defined through the grammar:

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

[17] This notion is also known as *lazy* reduction; for the sake of keeping our terminology consistent, we prefer to call it weak head reduction.

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

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 any term can be the body. Moreover, notice that both $\lambda z.\Delta\Delta$ and $\lambda z.\Omega\Omega$ are in WHNF.

Since $\to_{w\mathbf{x}\mathrm{H}} \subseteq \to_{\mathbf{x}\mathrm{H}}$, we can show the equivalent of Lem 1.8 and Theorem 7.3 also for *weak explicit head reduction*:

Proposition 8.4 *If $M \to^*_{\beta\mu} N$ with N in* WHNF*, then there exists $\boldsymbol{H}_w$ such that $M \to^{nf}_{w\mathrm{H}} \boldsymbol{H}_w$ and $\boldsymbol{H}_w \to^*_{\beta\mu} N$ without using $\to_{w\mathrm{H}}$.*

Corollary 8.5 *i) If $M \to^*_{w\mathbf{x}\mathrm{H}} N$, then $\llbracket M \rrbracket\, a \approx \llbracket N \rrbracket\, a$.*
ii) If $M\uparrow_{w\mathbf{x}\mathrm{H}}$, then $\llbracket M \rrbracket\, a \uparrow_\pi$.

We also define weak explicit head-normal forms.

Definition 8.6 (WEAK EXPLICIT HEAD-NORMAL FORMS FOR $\lambda\mu$) *i*) The $\lambda\mu\mathbf{x}$ *weak explicit head-normal forms (*WEHNF*)* are defined through the grammar:

$$\begin{array}{rcll} \boldsymbol{H}_{w\mathbf{x}} & ::= & (\lambda x.M)\,\overrightarrow{\langle y := N\rangle}\,\overrightarrow{\langle \sigma := Q\cdot\tau\rangle} & \\ & \mid & (xM_1\cdots M_n)\,\overrightarrow{\langle y := N\rangle}\,\overrightarrow{\langle \sigma := Q\cdot\tau\rangle} & (n \geq 0,\ x \notin \vec{y}) \\ & \mid & \mu\alpha.[\beta]\boldsymbol{H}_{w\mathbf{x}}\,\overrightarrow{\langle y := N\rangle}\,\overrightarrow{\langle \sigma := Q\cdot\tau\rangle} & (\alpha \notin \vec{\sigma},\ \alpha \neq \beta \text{ or } \alpha \in \boldsymbol{H}_{w\mathbf{x}}, \\ & & & \quad \text{and } \boldsymbol{H}_{w\mathbf{x}} \neq \mu\gamma.[\delta]\boldsymbol{H}'_{w\mathbf{x}}) \end{array}$$

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

Remark 8.7 In the context of reduction (normal and weak), when starting from pure terms, the substitution operation can be left inside terms in normal form, as in

$$(\lambda x.yM)NL \ \to_{\mathbf{x}\mathrm{H}}\ yM\,\langle x := N\rangle\, L.$$

However, since by Barendregt's convention we can assume that x does not appear free in L, the latter term is operationally equivalent to $yML\,\langle x := N\rangle$; in fact, these two are equivalent under $\sim_{w\mathrm{H}}$ (see Definition 9.3), and also congruent when interpreted as processes.

$$\begin{array}{rcll} \llbracket yM\langle x := N\rangle L \rrbracket\, a & \triangleq & (\nu c)\,((\nu x)(\llbracket yM \rrbracket\, c \mid \llbracket x := N \rrbracket) \mid \llbracket c := L\cdot a \rrbracket) & \equiv\ (c \notin \llbracket x := N \rrbracket, x \notin \llbracket c := L\cdot a \rrbracket) \\ & & (\nu x)((\nu c)\,(\llbracket yM \rrbracket\, c \mid \llbracket c := L\cdot a \rrbracket) \mid \llbracket x := N \rrbracket) & \triangleq \\ & & \llbracket yML\langle x := N\rangle \rrbracket\, a & \end{array}$$

Therefore, without loss of generality, for readability and ease of definition we will assume that all explicit substitutions are placed on the outside.[18] So actual terms can have substitutions inside, but they are written as if they appear outside. To ease notation, we will use $\boldsymbol{S}$ for a set of substitutions of the shape $\langle x := N\rangle$ or $\langle \alpha := N\cdot\gamma\rangle$ when the exact contents of the substitutions is not relevant. We write $x \in \boldsymbol{S}$ if $\langle x := N\rangle \in \boldsymbol{S}$ and similarly for $\alpha \in \boldsymbol{S}$.

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

Lemma 8.8 *If M has no* WEHNF *(so M also has no* WHNF*), then $\llbracket M \rrbracket\, a \approx \mathtt{0}$.*

[18] This is exactly the approach of Krivine's machine, where explicit substitutions are called *closures* that from an environment in which a term is evaluated.

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

We reason by co-induction on the explicit weak head reduction sequence from M and analyse the cases of weak explicit head reduction. We distinguish the cases:

$(M = (\mu\gamma.[\alpha_i]N)\,\langle\overrightarrow{y:=Q}\rangle\,\langle\overrightarrow{\alpha:=T\cdot\beta}\rangle \to \mu\gamma.[\alpha_i](N\langle\alpha_i:=T_i\cdot\beta_i\rangle\,T_i)\,\langle\overrightarrow{y:=Q}\rangle\,\langle\overrightarrow{\alpha:=T\cdot\beta}\rangle)$: Notice that:

$$\llbracket(\mu\gamma.[\alpha_i]N)\,\langle\overrightarrow{y:=Q}\rangle\,\langle\overrightarrow{\alpha:=T\cdot\beta}\rangle\rrbracket\, a \;\triangleq\; (\nu\vec{\alpha})((\nu\vec{y})\,(\llbracket N\rrbracket\,\alpha_i[a/\gamma] \mid \llbracket\overrightarrow{y:=Q}\rrbracket) \mid \llbracket\overrightarrow{\alpha:=T\cdot\beta}\rrbracket)$$

where

$$\begin{array}{rcl}\llbracket y_j:=Q_j\rrbracket &=& !\overline{y_j}(w).\llbracket Q_j\rrbracket\, w\\ \llbracket\alpha_k:=T_k\cdot\beta_k\rrbracket &=& !\alpha_k(v,d).(!\overline{v}(w).\llbracket T_k\rrbracket\, w \mid !d\rightarrow\beta_k)\end{array}$$

Observe that all inputs and outputs in these two categories are over bound names or under guard in $\llbracket M\rrbracket\, a$. Now, as in the proof of Theorem 7.1,

$$\begin{array}{ll}\llbracket(\mu\gamma.[\alpha_i]N)\langle\overrightarrow{y:=Q}\rangle\,\langle\overrightarrow{\alpha:=T\cdot\beta}\rangle\rrbracket\, a & \triangleq\\ (\nu\vec{\alpha})\,((\nu\vec{y})\,(\llbracket N\rrbracket\,\alpha_i[a/\gamma] \mid \llbracket\overrightarrow{y:=Q}\rrbracket) \mid \llbracket\overrightarrow{\alpha:=T\cdot\beta}\rrbracket) & \approx\\ (\nu\vec{\alpha})\,((\nu\vec{y})\,((\nu\alpha_i)\,(\llbracket N\rrbracket\,\alpha_i[a/\gamma] \mid \llbracket\alpha_i:=T_i\cdot\beta_i\rrbracket) \mid \llbracket\overrightarrow{y:=Q}\rrbracket) \mid \llbracket\overrightarrow{\alpha:=T\cdot\beta}\rrbracket) & \approx\\ (\nu\vec{\alpha})\,((\nu\vec{y})\,((\nu c)\,((\nu\alpha_i)(\llbracket N\rrbracket\, c \mid \llbracket\alpha_i:=T_i\cdot\beta_i\rrbracket) \mid \llbracket c:=T_i\cdot\beta_i\rrbracket)\,[a/\gamma] \mid \llbracket\overrightarrow{y:=Q}\rrbracket) \mid \llbracket\overrightarrow{\alpha:=T\cdot\beta}\rrbracket) & \triangleq\\ \llbracket\mu\gamma.[\alpha_i](N\,\langle\alpha_i:=T_i\cdot\beta_i\rangle\,T_i)\,\langle\overrightarrow{y:=Q}\rangle\,\langle\overrightarrow{\alpha:=T\cdot\beta}\rangle\rrbracket\, a & \end{array}$$

which, by co-induction, is weakly bisimilar to $\mathit{0}$; then so is $\llbracket M\rrbracket\, a$.

$(M = RP_1\cdots P_n\,\langle\overrightarrow{y:=Q}\rangle\,\langle\overrightarrow{\alpha:=T\cdot\beta}\rangle)$: Notice that:

$$\begin{array}{l}\llbracket RP_1\cdots P_n\,\langle\overrightarrow{y:=Q}\rangle\,\langle\overrightarrow{\alpha:=T\cdot\beta}\rangle\rrbracket\, a \;\triangleq\\ \qquad(\nu\vec{\alpha})\,((\nu\vec{y})\,((\nu\vec{c})\,(\llbracket R\rrbracket\, c_1 \mid \llbracket\overrightarrow{c_i:=P_i\cdot c_{i+1}}\rrbracket) \mid \llbracket\overrightarrow{y:=Q}\rrbracket) \mid \llbracket\overrightarrow{\alpha:=T\cdot\beta}\rrbracket)\end{array}$$

where $c_n = a$, $\llbracket y_j:=Q_j\rrbracket$ and $\llbracket\alpha_k:=T_k\cdot\beta_k\rrbracket$ are as above, and

$$\llbracket c_i:=P_i\cdot c_{i+1}\rrbracket \;=\; !c_i(v,d).(!\overline{v}(w).\llbracket P_i\rrbracket\, w \mid !d\rightarrow c_{i+1})$$

Again all inputs and outputs in these three categories are over bound names or under guard. Now either:

$(R = y_i)$: Then. as in the proof of Theorem 7.1, $\llbracket M\rrbracket\, a \approx$

$$(\nu\vec{\alpha})\,((\nu\vec{y})\,((\nu\vec{c})\,((\nu y_i)\,(y_i(u).!u\rightarrow c_1 \mid !\overline{y_i}(w).\llbracket Q_i\rrbracket\, w) \mid \llbracket\overrightarrow{c_i:=P_i\cdot c_{i+1}}\rrbracket) \mid \llbracket\overrightarrow{y:=Q}\rrbracket) \mid \llbracket\overrightarrow{\alpha:=T\cdot\beta}\rrbracket)$$

which after a synchronisation over y_i is weakly bisimilar to

$$(\nu\vec{\alpha})\,((\nu\vec{y})\,((\nu\vec{c})\,(\llbracket Q_i\rrbracket\, c_1 \mid \llbracket\overrightarrow{c_i:=P_i\cdot c_{i+1}}\rrbracket) \mid \llbracket\overrightarrow{y:=Q}\rrbracket) \mid \llbracket\overrightarrow{\alpha:=T\cdot\beta}\rrbracket)$$

which, by co-induction, is weakly bisimilar to $\mathit{0}$. Since y_i is restricted, so is $\llbracket M\rrbracket\, a$.

$(R = (\lambda x.K)L \to K\,\langle x:=L\rangle)$: Then:

$$\llbracket R\rrbracket\, c_1 \;\triangleq\; (\nu c_0)((\nu xb)(\llbracket K\rrbracket\, b \mid \overline{c_0}\langle x,b\rangle) \mid !c_0(v,d).(!\overline{v}(w).\llbracket L\rrbracket\, w \mid !d\rightarrow c_1))$$

Since a synchronisation over c_0 is possible, the process is not in normal form; also, since that name is restricted, only inputs or outputs generated by $(\nu xb)\,(\llbracket K\rrbracket\, b)$ can be observable. Executing that synchronisation will run to $\llbracket K\langle x:=L\rangle\rrbracket\, c_1 = (\nu x)(\llbracket K\rrbracket\, c_1 \mid !\overline{x}(w).\llbracket L\rrbracket\, w)$. By co-induction,

$$\llbracket K\,\langle x:=L\rangle\,P_1\cdots P_n\,\langle\overrightarrow{y:=Q}\rangle\,\langle\overrightarrow{\alpha:=T\cdot\beta}\rangle\rrbracket\, a \;\approx\; \mathit{0}$$

so also $(\nu xc_1)\,(\llbracket K\rrbracket\, c_1)$ is not producing input or output (remember that c_1 is restricted in

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

Figure 2: Running ${}^{\Vert}\Delta\Delta^{\Vert}\, a$ without renaming, but using garbage collection.

${}^{\Vert}M^{\Vert}\, a$), so neither does $(\nu xb)\,({}^{\Vert}K^{\Vert}\, b)$.

$(R = (\mu\alpha.[\beta]K)L \to \mu\gamma.([\beta]K)\,\langle\alpha := L{\cdot}\gamma\rangle)$: Then, as in the proof of Theorem 7.1 (remember that c is fresh),

$$
\begin{array}{rcl}
{}^{\Vert}R^{\Vert}\, c_1 & \stackrel{\Delta}{=} & (\nu c)\,({}^{\Vert}K^{\Vert}\, \beta[c/\alpha] \mid {}^{\Vert}c := L{\cdot}c_1{}^{\Vert}) \\
& =_{\alpha} & (\nu\alpha)({}^{\Vert}K^{\Vert}\, \beta \mid {}^{\Vert}\alpha{:=}L{\cdot}\gamma{}^{\Vert})\,[c_1/\gamma] \\
& \stackrel{\Delta}{=} & {}^{\Vert}\mu\gamma.([\beta]K)\,\langle\alpha := L{\cdot}\gamma\rangle{}^{\Vert}\, c_1
\end{array}
$$

As above, $(\nu\alpha)\,({}^{\Vert}\langle\alpha := L{\cdot}\gamma\rangle{}^{\Vert})$ creates no input or output, and by co-induction, $(\nu\alpha)\,{}^{\Vert}K^{\Vert}\, \beta$ does not produce input or output, so neither does $(\nu c)\,({}^{\Vert}K^{\Vert}\, \beta[c/\alpha])$.

$(R = \mu\alpha.[\beta](\mu\gamma.[\delta]K) \to \mu\alpha.[\delta]K[\beta/\gamma])$: Again, by the proof of Theorem 7.1,

$$
{}^{\Vert}\mu\alpha.[\beta](\mu\gamma.[\delta]K)^{\Vert}\, c_1 \;=\; {}^{\Vert}\mu\alpha.([\delta]K)[\beta/\gamma]^{\Vert}\, c_1
$$

so this case follows directly by co-induction.

$(R = \mu\alpha.[\alpha]K \to K\ (\alpha \notin fn(K)))$: As in the previous case. □

The reduction of ${}^{\Vert}\Delta\Delta^{\Vert}\, a$ is given in Figure 2, and shows that the interpretation of $\Delta\Delta$ reduces without creating output over a; notice that the individual steps of the above reduction in $\to_{\mathbf{xH}}$ in Example 5.5 are respected in Figure 2.

As a direct consequence of this result, as for Milner's and Sangiorgi's interpretations, our interpretation is not extensional, since ${}^{\Vert}\Delta\Delta^{\Vert}\, a \approx \mathit{0}$, whereas ${}^{\Vert}\lambda x.\Delta\Delta x^{\Vert}\, a \stackrel{\Delta}{=} (\nu xb)({}^{\Vert}\Delta\Delta x^{\Vert}\, b \mid \overline{a}\langle x,b\rangle) \not\approx \mathit{0}$.

We can show the following property.

Lemma 8.9 *i) Let M and N be pure $\lambda\mu$-terms; then $M \to^{nf}_{w\mathrm{H}} N$ if and only if there exists N', $\boldsymbol{S}$ such that $M \to^{nf}_{w\mathbf{x}\mathrm{H}} N'\,\boldsymbol{S}$, and $N'\,\boldsymbol{S} \to^{nf}_{:=} N$.*

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

Proof: Straightforward, using Corollary 8.5. □

We can also show the following results that state that if the interpretation of M produces an output, then M reduces by head reduction to an abstraction; similarly, if the interpretation of

M produces an input, then M reduces by head reduction to a term with a head variable.

Lemma 8.10 *i) If $M \to^{nf}_{w\mathbf{xH}} \lambda x.N\,\boldsymbol{S}$, then $\llbracket M \rrbracket\, a \Downarrow \overline{a}$.*

ii) If $M \to^{nf}_{w\mathbf{xH}} \mu\alpha.[\beta]\lambda x.N\,\boldsymbol{S}$, then $\llbracket M \rrbracket\, a \Downarrow \overline{\beta}$.

iii) If $M \to^{nf}_{w\mathbf{xH}} xN_1 \cdots N_n\,\boldsymbol{S}$ or $M \to^{nf}_{w\mathbf{xH}} \mu\alpha.[\beta]xN_1 \cdots N_n\,\boldsymbol{S}$, then $\llbracket M \rrbracket\, a \Downarrow x$.

Proof: Straightforward, using Theorem 7.3. □

As to the reverse, we can show:

Lemma 8.11 *i) If $\llbracket M \rrbracket\, a \Downarrow \overline{a}$, then there exist x, N and $\boldsymbol{S}$ such that $\llbracket M \rrbracket\, a \approx \llbracket \lambda x.N\,\boldsymbol{S} \rrbracket\, a$, such that $M \to^{nf}_{w\mathbf{xH}} \lambda x.N\,\boldsymbol{S}$.*

ii) If $\llbracket M \rrbracket\, a \Downarrow \overline{c}$, with $a \neq c$, then there exist α, c, x, N and $\boldsymbol{S}$ such that $\llbracket M \rrbracket\, a \approx \llbracket \mu\alpha.[c]\lambda x.N\,\boldsymbol{S} \rrbracket\, a$, such that $M \to^{nf}_{w\mathbf{xH}} \mu\alpha.[c]\lambda x.N\,\boldsymbol{S}$.

iii) If $\llbracket M \rrbracket\, a \Downarrow x$, then there exist $\vec{z_j}, x, \overrightarrow{N_i}, c$ and $\boldsymbol{S}$ with $x \notin \vec{z_j}$, $m \geq 0$, and $n \geq 0$ such that

- *$\llbracket M \rrbracket\, a \approx \llbracket \lambda z_1 \cdots z_m.xN_1 \cdots N_n\,\boldsymbol{S} \rrbracket\, c$;*
- *$M \to^{nf}_{w\mathbf{xH}} \lambda z_1 \cdots z_m.xN_1 \cdots N_n\,\boldsymbol{S}$ if $a = c$;*
- *$M \to^{nf}_{w\mathbf{xH}} \mu\alpha.[c]\lambda z_1 \cdots z_m.xN_1 \cdots N_n[a/\alpha]\,\boldsymbol{S}$, if $a \neq c$.*

Proof: By Theorem 7.4. □

9 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 $\sim_{w\mathbf{xH}}$ 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. This is also the case for the approximation semantics we present in the next section.

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

Definition 9.1 We define $\sim_{w\beta\mu}$ as the smallest congruence that contains:

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

Notice that $\Delta\Delta \sim_{w\beta\mu} \Omega\Omega$ 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. In Section 10 we will show that two terms are equivalent in $\sim_{w\beta\mu}$ if and only if they have the same set of weak approximants.

Since reduction is confluent, the following is immediate.

Proposition 9.2 *If $M \sim_{w\beta\mu} N$ and $M \to^*_{w\beta\mu} \boldsymbol{H}_w$, then there exists $\boldsymbol{H}'_w$ such that $\boldsymbol{H}_w \sim_{w\beta\mu} \boldsymbol{H}'_w$ and $N \to^*_{w\beta\mu} \boldsymbol{H}'_w$.*

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 9.6 that these coincide for pure, substitution-free terms.

Definition 9.3 (Weak head equivalence) The relation $\sim_{w\text{H}}$ is defined co-inductively as the largest symmetric relation such that: $M \sim_{w\text{H}} N$ if and only if either:

- M and N have both no WHNF, or
- both $M \to^{nf}_{w\text{H}} M'$ and $N \to^{nf}_{w\text{H}} N'$, and either:
  - if $M' = xM_1 \cdots M_n$ $(n \geq 0)$, then $N' = xN_1 \cdots N_n$ and $M_i \sim_{w\text{H}} N_i$ for all $i \in \underline{n}$; or
  - if $M' = \lambda x.M''$, then $N' = \lambda x.N''$ and $M'' \sim_{w\text{H}} N''$; 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 8.3, 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 9.4 (Weak explicit head equivalence) The relation $\sim_{w\text{xH}}$ is defined co-inductively as the largest symmetric relation 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 $M \to^{nf}_{w\text{xH}} M'\boldsymbol{S}$ and $N \to^{nf}_{w\text{xH}} N'\boldsymbol{S}'$, and either:
  - if $M' = xM_1 \cdots M_n$ $(n \geq 0)$, then $N' = xN_1 \cdots N_n$ (so $x \notin \boldsymbol{S}$, $x \notin \boldsymbol{S}'$) and $M_i\boldsymbol{S} \sim_{w\text{xH}} N_i\boldsymbol{S}'$ for all $i \in \underline{n}$; or
  - if $M' = \lambda x.M''$, then $N' = \lambda x.N''$ and $M''\boldsymbol{S} \sim_{w\text{xH}} N''\boldsymbol{S}'$; or
  - if $M' = \mu\alpha.[\beta]M''$, then $N' = \mu\alpha.[\beta]N''$ (so $\alpha \neq \beta$ or $\alpha \in fn(M'')$, $M'' \neq \mu\gamma.[\delta]R$, so $\beta \notin \boldsymbol{S}$, $\beta \notin \boldsymbol{S}'$, and similarly for N'') and $M''\boldsymbol{S} \sim_{w\text{xH}} N''\boldsymbol{S}'$.

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 9.5 Let M and N be pure $\lambda\mu$-terms. $M \sim_{w\text{H}} N$ if and only if there are M', N' such that $M' \to^{nf}_{:=} M$ and $N' \to^{nf}_{:=} N$, and $M' \sim_{w\text{xH}} N'$.

Proof: (only if): By co-induction on the definition of $\sim_{w\text{H}}$. If $M \sim_{w\text{H}} N$, then either:

- *$M \to^{nf}_{w\text{H}} xM_1 \cdots M_n$ and $N \to^{nf}_{w\text{H}} xN_1 \cdots N_n$ and $M_i \sim_{w\text{H}} N_i$, for all $i \in \underline{n}$. Then, by Lemma 8.9, there are $\overrightarrow{M'_i}$ such that both*

$$\begin{array}{lllll} M & \to^{nf}_{w\text{xH}} & xM'_1 \cdots M'_n\,\boldsymbol{S} & \to^{*}_{:=} & xM_1 \cdots M_n \\ N & \to^{nf}_{w\text{xH}} & xN'_1 \cdots N'_n\,\boldsymbol{S}' & \to^{*}_{:=} & xN_1 \cdots N_n \end{array}$$

But then $M'_i\boldsymbol{S} \to^{nf}_{:=} M_i$ and $N'_i\boldsymbol{S}' \to^{nf}_{:=} N_i$, for all $i \in \underline{n}$; then by induction, $M'_i\boldsymbol{S} \sim_{w\text{xH}} N'_i\boldsymbol{S}'$ for all $i \in \underline{n}$. But then $M \sim_{w\text{xH}} N$.

The other cases are similar.

(*if*): *By co-induction on the definition of* $\sim_{w\mathbf{xH}}$. *If there are* M', N' *such that* $M' \to^{nf}_{:=} M$ *and* $N' \to^{nf}_{:=} N$, *and* $M' \sim_{w\mathbf{xH}} N'$, *then either:*

- $M' \to^{nf}_{w\mathbf{xH}} xM'_1 \cdots M'_n\, \boldsymbol{S}$, $N' \to^{nf}_{w\mathbf{xH}} xN'_1 \cdots N'_n\, \boldsymbol{S}'$ *and* $M'_i \boldsymbol{S} \sim_{w\mathbf{xH}} N'_i \boldsymbol{S}'$, *for all* $i \in \underline{n}$. *Let, for all* $i \in \underline{n}$, $M'_i \boldsymbol{S} \to^{nf}_{:=} M_i$ *and* $N'_i \boldsymbol{S} \to^{nf}_{:=} N_i$ *then by induction,* $M_i \sim_{w\mathbf{H}} N_i$, *for all* $i \in \underline{n}$.

  Notice that we have $M' \to^{nf}_{w\mathbf{xH}} xM'_1 \cdots M'_n\, \boldsymbol{S} \to^{nf}_{:=} xM_1 \cdots M_n$. *Let* $M' = M''\, \boldsymbol{S}''$, *so* $M''\, \boldsymbol{S}'' \to^{nf}_{w\mathbf{xH}} xM'_1 \cdots M'_n\, \boldsymbol{S}'\, \boldsymbol{S}''$, *where* $\boldsymbol{S} = \boldsymbol{S}'\, \boldsymbol{S}''$. *Let* $M''\, \boldsymbol{S}'' \to^{nf}_{:=} M$, *then by Lemma 8.9, we also have* $M \to^{nf}_{w\mathbf{xH}} xM''_1 \cdots M''_n\, \boldsymbol{S}' \to^{nf}_{w\mathbf{H}} xM_1 \cdots M_n$. *Then, again by Lemma 8.9,* $M \to^{nf}_{w\mathbf{H}} xM_1 \cdots M_n$.

  Likewise, we have $N \to^{nf}_{w\mathbf{H}} xN_1 \cdots N_n$. *But then* $M \sim_{w\mathbf{H}} N$.

The other cases are similar. □

Notice that this lemma in fact shows:

Theorem 9.6 *Let* $M, N \in \lambda\mu$, *then* $M \sim_{w\mathbf{xH}} N \iff M \sim_{w\mathbf{H}} N$.

10 Weak approximation for $\lambda\mu$

In the next section we will show our main result, that the logical encoding is fully abstract with respect to weak equivalence between pure $\lambda\mu$-terms. To achieve this, we show in Theorem 11.1 that $\llbracket M \rrbracket\, a \approx \llbracket N \rrbracket\, a$ *if and only if* $M \sim_{w\mathbf{xH}} N$. To complete the proof towards $\sim_{w\beta\mu}$, we are thus left with the obligation to show that $M \sim_{w\mathbf{xH}} N$ *if and only if* $M \sim_{w\beta\mu} N$. In Theorem 9.6 we have shown that $M \sim_{w\mathbf{xH}} N$ *if and only if* $M \sim_{w\mathbf{H}} N$, for pure terms; to achieve $M \sim_{w\mathbf{H}} N$ *if and only if* $M \sim_{w\beta\mu} N$, in this section we go through a notion of *weak approximation*; based on Wadsworth's approach [48], we define $\sim_{\mathcal{A}_w}$ that expresses that terms have the same weak approximants and show that $M \sim_{w\mathbf{H}} N$ *if and only if* $M \sim_{\mathcal{A}_w} N$ *if and only if* $M \sim_{w\beta\mu} N$.

The notions of *approximant* and *approximation* were first introduced by Wadsworth for the λ-calculus [48], 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 $\lambda x.yN_1 \cdots N_n$, all remaining redexes occur in $N_1, \ldots, N_n$, which then in turn will be approximated.

Following this approach, Wadsworth [48] defines $\mathcal{A}(M)$ (similar to Definition 10.1 below) as the set of approximants of the λ-term M, which forms a meet semi-lattice. In [49], the connection is established between approximation and semantics, by showing

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

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 [46, 23]; however, seen that we are mainly interested in *weak* reduction here, we will define *weak* approximants, which are normally not considered.

Definition 10.1 (Weak approximation for $\lambda\mu$) *i)* The set of $\lambda\mu$'s *weak approximants* $\mathcal{A}_w$ with respect to $\rightarrow_{\beta\mu}$ is defined through the grammar:[19]

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

ii) The relation $\sqsubseteq \subseteq (\lambda\mu \cup \{\bot\}) \times \lambda\mu$ is defined as:[20]

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

iii) The set of *weak approximants* of M, $\mathcal{A}_w(M)$, is defined through:[21]

$$\mathcal{A}_w(M) \ \stackrel{\Delta}{=}\ \{\, \mathbf{A}_w \in \mathcal{A}_w \mid \exists N \in \lambda\mu\ [M \rightarrow^*_{\beta\mu} N \ \&\ \mathbf{A}_w \sqsubseteq N]\,\}.$$

iv) *Weak approximation equivalence* is defined through: $M \sim_{\mathcal{A}_w} N \ \stackrel{\Delta}{=}\ \mathcal{A}_w(M) = \mathcal{A}_w(N)$.

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

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

Lemma 10.2 *i) If $\mathbf{A}_w \sqsubseteq M$ and $M \rightarrow^*_{\beta\mu} N$, then $\mathbf{A}_w \sqsubseteq N$.*

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

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

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

Proof: Easy. □

We could also have defined the set of approximants of a term coinductively:

Definition 10.3 We define $\mathcal{A}^w(M)$ coinductively by:

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

We can show that these definitions coincide:

Lemma 10.4 $\mathcal{A}^w(M) = \mathcal{A}_w(M)$.

Proof: $(\subseteq)$: If $\mathbf{A}_w \in \mathcal{A}^w(M)$, then by Definition 10.3 either:

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

---

[19] For 'normal' approximants, case $\lambda x.A$ demands that $A \neq \bot$, as motived by the relation with D_∞.

[20] Notice that if $A_1 \sqsubseteq M_1$, and $A_2 \sqsubseteq M_2$, then A_1A_2 need not be an approximant; it is one if $A_1 = xA_1^1\cdots A_1^n$, perhaps prefixed with a number of context switches of the shape $\mu\alpha.[\beta]$.

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

$(\mathbf{A}_w = x\mathbf{A}_w^1 \cdots \mathbf{A}_w^n)$: Then $M \to_{w\mathrm{H}}^* xM_1 \cdots M_n$ for some $M_1, \ldots M_n$, with $\mathbf{A}_w^i \in \mathcal{A}^w(M_i)$, for every $i \in \underline{n}$; by co-induction, also $\mathbf{A}_w^i \in \mathcal{A}_w(M_i)$. Then, by Definition 10.1, for every $i \in \underline{n}$ there exist M_i' such that $M_i \to_{\beta\mu}^* M_i'$ and $\mathbf{A}_w^i \sqsubseteq M_i'$. Since $\to_{w\mathrm{H}}^* \subseteq \to_{\beta\mu}^*$, in particular $M \to_{\beta\mu}^* xM_1' \cdots M_n'$; we have $\mathbf{A}_w \sqsubseteq xM_1' \cdots M_n'$, so $\mathbf{A}_w \in \mathcal{A}_w(M)$.

The other cases are similar.

$(\supseteq)$: If $\mathbf{A}_w \in \mathcal{A}_w(M)$, then by Definition 10.1, there exists N such that $M \to_{\beta\mu}^* N$ and $\mathbf{A}_w^i \sqsubseteq N$. Now either:

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

$(\mathbf{A}_w = x\mathbf{A}_w^1 \cdots \mathbf{A}_w^n)$: Since $x\mathbf{A}_w^1 \cdots \mathbf{A}_w^n \sqsubseteq N$, $N = xN_1 \cdots N_n$ for some $N_1, \ldots, N_n$, and $\mathbf{A}_w^i \sqsubseteq N_i$, for every $i \in \underline{n}$. Then by Definition 10.3, $\mathbf{A}_w^i \in \mathcal{A}^w(N_i)$, for every $i \in \underline{n}$, and by induction, $\mathbf{A}_w^i \in \mathcal{A}_w(N_i)$. By Lemma 8.4, there exist $M_1, \ldots, M_n$ such that $M \to_{w\mathrm{H}}^* xM_1 \cdots M_n \to_{\beta\mu}^* xN_1 \cdots N_n$; so in particular $M_i \to_{\beta\mu}^* N_i$, for every $i \in \underline{n}$. Then by Lemma 10.2, $\mathbf{A}_w^i \in \mathcal{A}_w(M_i)$ and by Definition 10.3, $\mathbf{A}_w \in \mathcal{A}^w(M)$.

The other cases are similar. □

As a result, below we will use whichever definition is convenient.

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

Theorem 10.5 (Weak approximation semantics) *If $M =_{\beta\mu} N$, then $M \sim_{\mathcal{A}_w} N$.*

Proof:

$$\begin{array}{lll} M =_{\beta\mu} N \;\&\; \mathbf{A}_w \in \mathcal{A}_w(M) & \Rightarrow & \\ M =_{\beta\mu} N \;\&\; \exists L\, [M \to_{\beta\mu}^* L \;\&\; \mathbf{A}_w \sqsubseteq L] & \Rightarrow & (1.5) \\ \exists L, K\, [L \to_{\beta\mu}^* K \;\&\; N \to_{\beta\mu}^* K \;\&\; \mathbf{A}_w \sqsubseteq L] & \Rightarrow & (10.2) \\ \exists K\, [N \to_{\beta\mu}^* K \;\&\; \mathbf{A}_w \sqsubseteq K] & \Rightarrow & \mathbf{A}_w \in \mathcal{A}_w(N) \end{array}$$

□

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

Theorem 10.6 (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 co-induction on the definition of the set of weak approximants. If $\mathcal{A}_w(M) = \{\perp\} = \mathcal{A}_w(N)$, then both M and N have no whnf, so $M \sim_{w\beta\mu} N$. Otherwise, either:

$(x\mathbf{A}_w^1 \cdots \mathbf{A}_w^n \in \mathcal{A}_w(M) \;\&\; x\mathbf{A}_w^1 \cdots \mathbf{A}_w^n \in \mathcal{A}_w(N))$: Then by Definition 10.3 there exists $M_1, \ldots, M_n$ such that $M \to_{w\mathrm{H}}^* xM_1 \cdots M_n$ and $\mathbf{A}_w^i \in \mathcal{A}_w(M_i)$. Likewise, there exist $N_1, \ldots, N_n$ such that $N \to_{w\mathrm{H}}^* xN_1 \cdots N_n$ and $\mathbf{A}_w^i \in \mathcal{A}_w(N_i)$. So $\mathcal{A}_w(M_i) = \mathcal{A}_w(N_i)$ and by induction $M_i \sim_{w\beta\mu} N_i$, for $i \in \underline{n}$. Since $\sim_{w\beta\mu}$ is a congruence, also $xM_1 \cdots M_n \sim_{w\beta\mu} xN_1 \cdots N_n$; since $\sim_{w\beta\mu}$ is closed under reduction $\to_{w\beta\mu}$, it is also under $\to_{w\mathrm{H}}$, and we have $M \sim_{w\beta\mu} N$.

The other cases are similar.

(only if): As the proof of Theorem 10.5, but using Proposition 9.2 rather than 1.5. □

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

Theorem 10.7 *$M \sim_{w\mathrm{H}} N$ if and only if $M \sim_{\mathcal{A}_w} N$.*

Proof: *(only)*: By co-induction on the definition of $\sim_{w\mathrm{H}}$.

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

$(M \to^*_{w\text{H}} xM_1 \cdots M_n)$: Then also $N \to^*_{w\text{H}} xN_1 \cdots N_n$, and $M_i \sim_{w\text{H}} N_i$ for $i \in \underline{n}$, and by co-induction, $M_i \sim_{\mathcal{A}_w} N_i$, so $\mathcal{A}_w(M_i) = \mathcal{A}_w(N_i)$. Then, by Definition 10.3, we have $\mathcal{A}_w(M) = \mathcal{A}_w(N)$.

The other cases are similar.

(if): By co-induction on the definition of the set of weak approximants.

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

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

The other cases are similar. □

Taking $\sqcup$ as the (partial, compatible) operation of join on terms in $\mathcal{A}_w$ generated by $\bot \sqcup \mathbf{A}_w = \mathbf{A}_w$, we can also define $\llbracket M \rrbracket_{\mathcal{A}_w} = \sqcup\{\, \mathbf{A}_w \mid \mathbf{A}_w \in \mathcal{A}_w(M) \,\}$; then $\llbracket \cdot \rrbracket_{\mathcal{A}_w}$ corresponds to the ($\lambda\mu$ variant of) Lévy-Longo trees.

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

Corollary 10.8 *Let $M, N \in \lambda\mu$, then $M \sim_{w\text{XH}} N \iff M \sim_{w\text{H}} N \iff M \sim_{\mathcal{A}_w} N \iff M \sim_{w\beta\mu} N$.*

11 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 11.1 (FULL ABSTRACTION OF $\approx$ VERSUS $\sim_{w\text{XH}}$) *For any $M, N \in \lambda\mu\mathbf{x}$: $\llbracket M \rrbracket\, a \approx \llbracket N \rrbracket\, a$ if and only if $M \sim_{w\text{XH}} N$.*

Proof: $(\mathit{only\ if})$: We distinguish the following cases.

a) $\llbracket M \rrbracket\, a$ can never input nor output; then $\llbracket M \rrbracket\, a \approx \mathit{0} \approx \llbracket N \rrbracket\, a$. Assume M has a weak-head normal form, then by Lemma 8.10, $\llbracket M \rrbracket\, a$ is not weakly bisimilar to $\mathit{0}$; therefore, M and N both have no weak-head normal form.

b) $\llbracket M \rrbracket\, a \Downarrow \overline{c}$, then by Lemma 8.11, $\llbracket M \rrbracket\, a \approx (\nu xb)\,(\llbracket M' \rrbracket\, b \mid \overline{c}\langle x,b\rangle \mid \llbracket \boldsymbol{S} \rrbracket)$, and $M \to^*_{w\text{XH}} \lambda x.M'\,\boldsymbol{S}$. Since $\llbracket M \rrbracket\, a \approx \llbracket N \rrbracket\, a$, also $\llbracket N \rrbracket\, a \Downarrow \overline{c}$, so $\llbracket N \rrbracket\, a \approx (\nu xb)\,(\llbracket N' \rrbracket\, b \mid \overline{c}\langle x,b\rangle \mid \llbracket \boldsymbol{S}' \rrbracket)$ and $N \to^*_{w\text{XH}} \lambda x.N'\,\boldsymbol{S}'$. Then also $\llbracket M' \rrbracket\, b \mid \llbracket \boldsymbol{S} \rrbracket \approx \llbracket N' \rrbracket\, b \mid \llbracket \boldsymbol{S}' \rrbracket$, so $\llbracket M'\,\boldsymbol{S} \rrbracket\, a \approx \llbracket N'\,\boldsymbol{S}' \rrbracket\, a$ and by induction, $M'\,\boldsymbol{S} \sim_{w\text{XH}} N'\,\boldsymbol{S}'$; so also $M \sim_{w\text{XH}} N$ by definition.

c) If $\llbracket M \rrbracket\, a \not\Downarrow_o$, but $\llbracket M \rrbracket\, a \Downarrow x$, then by Lemma 8.11, $\llbracket M \rrbracket\, a \approx \llbracket xM_1 \cdots M_n\,\boldsymbol{S} \rrbracket\, a'$ and $M \to^*_{w\text{XH}} xM_1 \cdots M_n\,\boldsymbol{S}$. We have

$$\llbracket xM_1 \cdots M_n\,\boldsymbol{S} \rrbracket\, a' \;=\; (\nu \overline{cy\alpha})\,(x(u).!\,u \to c_1 \mid \llbracket \overrightarrow{c_i := M_i \cdot c_{i+1}} \rrbracket \mid \llbracket \boldsymbol{S} \rrbracket)$$

where $c_n = a'$ and

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

Since $\llbracket M \rrbracket\, a \approx \llbracket N \rrbracket\, a$, again by Lemma 8.11, $\llbracket N \rrbracket\, a \approx \llbracket xN_1 \cdots N_n\,\boldsymbol{S}' \rrbracket\, a''$ and $N \to^*_{w\text{XH}}$

$xN_1 \cdots N_n \boldsymbol{S}'$. Notice that

$$\llbracket xN_1 \cdots N_n \boldsymbol{S}' \rrbracket\, a'' \;=\; (\nu \overrightarrow{e y \alpha})\,(x(u).!\,u \rightarrow e_1 \mid \llbracket \overrightarrow{e_i := N_i \cdot e_{i+1}} \rrbracket \mid \llbracket \boldsymbol{S}' \rrbracket)$$

where $e_n = a''$ and

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

Since we have

$$\llbracket xM_1 \cdots M_n \boldsymbol{S} \rrbracket\, a' \;\approx\; \llbracket xN_1 \cdots N_n \boldsymbol{S}' \rrbracket\, a'',$$

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

(if): By co-induction on the definition of $\sim_{w\mathrm{xH}}$. Let $M \sim_{w\mathrm{xH}} N$, then either:

- M and N have both no $\rightarrow_{w\mathrm{xH}}$-normal form, so, by Lemma 8.8, their interpretations are both weakly bisimilar to the process $\mathit{0}$, so in particular $\llbracket M \rrbracket\, a \approx \llbracket N \rrbracket\, a$; or
- both $M \rightarrow^{nf}_{w\mathrm{xH}} M' \boldsymbol{S}$ and $N \rightarrow^{nf}_{w\mathrm{xH}} N' \boldsymbol{S}'$ (let $\boldsymbol{S} = \langle \overrightarrow{y := P} \rangle\, \langle \overrightarrow{\alpha := Q \cdot \beta} \rangle$, $\boldsymbol{S}' = \langle \overrightarrow{y := P'} \rangle\, \langle \overrightarrow{\alpha := Q' \cdot \beta} \rangle$),[22] and either:

$(M' = xM_1 \cdots M_n\ (n \geq 0),\ N = xN_1 \cdots N_n$ *and* $M_i \boldsymbol{S} \sim_{w\mathrm{xH}} N_i \boldsymbol{S}'$, *for all* $i \in \underline{n})$: By Corollary 8.5, we know that both $\llbracket M \rrbracket\, a \approx \llbracket xM_1 \cdots M_n \boldsymbol{S} \rrbracket\, a$ and $\llbracket N \rrbracket\, a \approx \llbracket xN_1 \cdots N_n \boldsymbol{S}' \rrbracket\, a$. Notice that

$$\llbracket xM_1 \cdots M_n \boldsymbol{S} \rrbracket\, a \;=\; (\nu \overrightarrow{c y \alpha})\,(x(u).!\,u \rightarrow c_1 \mid \llbracket \overrightarrow{c_i := M_i \cdot c_{i+1}} \rrbracket \mid \llbracket \boldsymbol{S} \rrbracket)$$

where $c_n = a$ and

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

and similar for $\llbracket xN_1 \cdots N_n \boldsymbol{S}' \rrbracket\, a$. By induction,

$$(\nu \overrightarrow{y \alpha})\,(\llbracket M_i \rrbracket\, w \mid \llbracket \boldsymbol{S} \rrbracket) \;\stackrel{\Delta}{=}\; \llbracket M_i \boldsymbol{S} \rrbracket\, w \;\approx\; \llbracket N_i \boldsymbol{S}' \rrbracket\, w \;\stackrel{\Delta}{=}\; (\nu \overrightarrow{y \alpha})\,(\llbracket N_i \rrbracket\, w \mid \llbracket \boldsymbol{S}' \rrbracket)$$

Since $\approx$ is a congruence, also

$$!\,c_i(v,d).(!\,\overline{v}(w).\llbracket M_i \rrbracket\, w \mid !\,d \rightarrow c_{i+1}) \mid \llbracket \boldsymbol{S} \rrbracket \;\approx\; !\,c_i(v,d).(!\,\overline{v}(w).\llbracket N_i \rrbracket\, w \mid !\,d \rightarrow c_{i+1}) \mid \llbracket \boldsymbol{S}' \rrbracket$$

for all $i \in \underline{n}$, so also $\llbracket xM_1 \cdots M_n \boldsymbol{S} \rrbracket\, a \approx \llbracket xN_1 \cdots N_n \boldsymbol{S}' \rrbracket\, a$ but then also $\llbracket M \rrbracket\, a \approx \llbracket N \rrbracket\, a$.

$(M' = \lambda x.M'',\ N' = \lambda x.N''$, *and* $M'' \boldsymbol{S} \sim_{w\mathrm{xH}} N'' \boldsymbol{S}')$: By Corollary 8.5, we have $\llbracket M \rrbracket\, a \approx \llbracket \lambda x.M'' \boldsymbol{S} \rrbracket\, a$ and $\llbracket N \rrbracket\, a \approx \llbracket \lambda x.N'' \boldsymbol{S}' \rrbracket\, a$. Notice that

$$\begin{array}{rcl}
\llbracket \lambda x.M'' \boldsymbol{S} \rrbracket\, a & \stackrel{\Delta}{=} & (\nu \overrightarrow{y \alpha})\,((\nu x b)(\llbracket M'' \rrbracket\, b \mid \overline{a}\langle x,b \rangle) \mid \llbracket \boldsymbol{S} \rrbracket) \\
\llbracket \lambda x.N'' \boldsymbol{S}' \rrbracket\, a & \stackrel{\Delta}{=} & (\nu \overrightarrow{y \alpha})\,((\nu x b)(\llbracket N'' \rrbracket\, b \mid \overline{a}\langle x,b \rangle) \mid \llbracket \boldsymbol{S}' \rrbracket)
\end{array}$$

with $\boldsymbol{S}$ and $\boldsymbol{S}'$ as in the previous part and a not in $\boldsymbol{S}$ or $\boldsymbol{S}'$. By induction,

$$(\nu \overrightarrow{y \alpha})\,(\llbracket M'' \rrbracket\, b \mid \llbracket \boldsymbol{S} \rrbracket) \;\stackrel{\Delta}{=}\; \llbracket M'' \boldsymbol{S} \rrbracket\, b \;\approx\; \llbracket N'' \boldsymbol{S}' \rrbracket\, b \;\stackrel{\Delta}{=}\; (\nu \overrightarrow{y \alpha})\,(\llbracket N'' \rrbracket\, b \mid \llbracket \boldsymbol{S} \rrbracket)$$

[22] Formally, the substitutions $\boldsymbol{S}$ and $\boldsymbol{S}'$ need not concern exactly the same variables and names; however, extending both so that they do does not affect the result, and gives an easier presentation of the proof.

As above, since $\approx$ is a congruence, also $\llbracket M \rrbracket\, a \approx \llbracket N \rrbracket\, a$.

$(M' = \mu\gamma.[\delta]M'',\ N' = \mu\gamma.[\delta]N'')$: Then M'' and N'' themselves are in normal form and $M''\,\boldsymbol{S} \sim_{w\mathbf{x}\mathrm{H}} N''\,\boldsymbol{S}'$, with $\boldsymbol{S}$, $\boldsymbol{S}'$ as above. By Corollary 8.5, $\llbracket M \rrbracket\, a \approx \llbracket \mu\gamma.[\delta]M''\,\boldsymbol{S} \rrbracket\, a$ and $\llbracket N \rrbracket\, a \approx \llbracket \mu\gamma.[\delta]N''\,\boldsymbol{S}' \rrbracket\, a$. Notice that

$$\begin{aligned}
\llbracket \mu\gamma.[\delta].M''\,\boldsymbol{S} \rrbracket\, a &\triangleq (\nu\overrightarrow{y\alpha})\,(\llbracket M''[a/\gamma] \rrbracket\, \delta \mid \llbracket \boldsymbol{S} \rrbracket) \triangleq \llbracket M''[a/\gamma]\,\boldsymbol{S} \rrbracket\, \delta \\
\llbracket \mu\gamma.[\delta].N''\,\boldsymbol{S}' \rrbracket\, a &\triangleq (\nu\overrightarrow{y\alpha})\,(\llbracket N''[a/\gamma] \rrbracket\, \delta \mid \llbracket \boldsymbol{S}' \rrbracket) \triangleq \llbracket N''[a/\gamma]\,\boldsymbol{S}' \rrbracket\, \delta
\end{aligned}$$

By induction, $\llbracket M''[a/\gamma]\,\boldsymbol{S} \rrbracket\, \delta \approx \llbracket N''[a/\gamma]\,\boldsymbol{S}' \rrbracket\, \delta$; since $\approx$ is a congruence, also $\llbracket M \rrbracket\, a \approx \llbracket N \rrbracket\, a$. □

We can now prove our main result:

Theorem 11.2 (Full abstraction) *Let $M, N \in \lambda\mu$, then $\llbracket M \rrbracket\, a \approx \llbracket N \rrbracket\, a$ if and only if $M \sim_{w\beta\mu} N$.*

Proof: By Corollary 10.8 and Theorem 11.1. □

Conclusions and Future Work

We defined $\lambda\mu\mathbf{x}$, a variant of $\lambda\mu$ that uses explicit substitution, and defined a notion of explicit head reduction $\rightarrow_{\mathbf{x}\mathrm{H}}$ 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 names the anonymous output of terms and respects $\rightarrow_{\mathbf{x}\mathrm{H}}$. 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 have shown that, for our context assignment system that uses the type constructor $\rightarrow$ for π and is based on classical logic, typeable $\lambda\mu$-terms are interpreted by our interpretation as typeable π-processes, preserving the types.

We also defined a weak variant of explicit head reduction, $\rightarrow_{w\mathbf{x}\mathrm{H}}$. 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 $\sim_{w\beta\mu}$, $\sim_{w\mathrm{H}}$, $\sim_{w\mathbf{x}\mathrm{H}}$, and $\sim_{\mathcal{A}_w}$ on $\lambda\mu$ terms, and have shown that these all coincide on *pure* terms (without explicit substitution). We then proved that $M \sim_{w\mathbf{x}\mathrm{H}} N \iff \llbracket M \rrbracket\, a \approx \llbracket N \rrbracket\, a$, which, combined with our other results, shows that our interpretation is fully abstract.

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, 1994.

[7] 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.

[8] S. van Bakel, L. Cardelli, and M.G. Vigliotti. Classical Cut-elimination in the π-calculus. Submitted, 2014.

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

[10] 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.

[11] 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, September 26-28 2012.

[12] S. van Bakel and M.G. Vigliotti. An Implicative Logic based encoding of the λ-calculus into the π-calculus. Submitted, 2014.

[13] S. van Bakel and M.G. Vigliotti. A fully abstract semantics of $\lambda\mu$ in the π-calculus. In *Electronic Proceedings of Sixth International Workshop on Classical Logic and Computation 2008* (CL&C'14), *Vienna, Austria*, volume 194 of *Electronic Proceedings in Theoretical Computer Science*, pages 33–47, 2014.

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

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

[23] U. de'Liguoro. The Approximation Theorem for the $\Lambda\mu$-Calculus. To appear in MSCS, 2014.

[24] G. Gentzen. Investigations into logical deduction. In *The Collected Papers of Gerhard Gentzen*. Ed M. E. Szabo, North Holland, 68ff (1969), 1935.

[25] J.-Y Girard. Linear Logic. *Theoretical Computer Science*, 50:1–102, 1987.

[26] 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.

[27] H. Herbelin. *C'est maintenant qu'on calcule: au cœur de la dualité*. Mémoire d'habilitation, Université Paris 11, Décembre 2005.

[28] 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.

[29] 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, July 15-19, 1991, Proceedings*, volume 512 of *Lecture Notes in Computer Science*, pages 133–147. Springer Verlag, 1991.

[30] K. Honda, N. Yoshida, and M. Berger. Control in the π-Calculus. In *Proceedings of Fourth ACM-SIGPLAN Continuation Workshop* (CW'04), 2004.

[31] Kohei Honda. Types for Dyadic Interaction. In Eike Best, editor, *CONCUR '93, 4th International Conference on Concurrency Theory, Hildesheim, Germany, August 23-26, 1993, Proceedings*, volume 715 of *Lecture Notes in Computer Science*, pages 509–523. Springer Verlag, 1993.

[32] 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.

[33] J-L. Krivine. A call-by-name lambda-calculus machine. *Higher Order and Symbolic Computation*, 20(3):199–207, 2007.

[34] 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), 12-15 August 2006, Seattle, WA, USA*, pages 297–306, 2006.

[35] 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.

[36] G. Longo. Set-theoretical models of λ-calculus: theories, expansions and isomorphisms. *Annals of Pure and Applied Logic*, 24(2):153–188, 1983.

[37] R. Milner. Functions as Processes. *Mathematical Structures in Computer Science*, 2(2):269–310, 1992.

[38] 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, Paris (France)*, pages 215–227, 1997.

[39] 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.

[40] M. Parigot. Strong Normalization for Second Order Classical Natural Deduction. In *Proceedings of Eighth Annual IEEE Symposium on Logic in Computer Science, 19-23 June 1993, Montreal, Canada*, pages 39–46, 1993.

[41] B.C. Pierce and D. Sangiorgi. Typing and Subtyping for Mobile Processes. *Mathematical Structures in Computer Science*, 6(5):409–453, 1996.

[42] W. Py. *Confluence en $\lambda\mu$-calcul*. Phd thesis, Université de Savoie, 1998.

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

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

[45] 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.

[46] 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, April 19-21 2010.

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

[48] 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.

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