An output-based semantic interpretation of λ in π
(Extended Abstract)

Steffen van Bakel and Maria Grazia Vigliotti

Department of Computing, Imperial College London,
180 Queen's Gate, London SW7 2BZ, UK
{s.vanbakel,maria.vigliotti}@imperial.ac.uk

Abstract. We define a compositional *output*-based interpretation of the λ-calculus with explicit substitution into a variant of the π-calculus with pairing, and show that this interpretation preserves *full* single-step β-reduction (*i.e.* not just open applicative bisimilarity) with respect to contextual equivalence. For this interpretation, we show the customary operational soundness for β-reduction, adequacy, and operational completeness; using a notion of implicative type-context assignment for the π-calculus, we also show that assignable Curry types are preserved by the interpretation. We finish by showing that termination is preserved for spine reduction with respect to lazy reduction for the π-calculus.

Introduction

The π-calculus and its dialects have proven to give an interesting model of computation. Encoding of variants of pure [19, 24, 23, 5] and typed [17] λ-calculus [9, 7] and object oriented calculi [15, 24] have been shown. Also, various encodings of calculi that represent classical logic have been recently proposed [17, 4, 10]. In this paper we investigate the expressive power of the asynchronous π-calculus [19] extended with pairing by showing a new compositional semantic interpretation of $\lambda\mathbf{x}$, the explicit substitution λ-calculus [1, 8] that fully respects *full single-step reduction*; through this result we will show that our interpretation gives a semanticsfor the λ-calculus.

The advantage of considering the explicit substitution λ-calculus rather than the standard implicit substitution of the λ-calculus as considered in [24] was already argued in [5]. In that work we showed that communication in the π-calculus has a fine semantic level of granularity that 'faithfully mimics' explicit substitution, rather than implicit substitution; we stress this point again with the results presented in this paper. Our interpretation encodes not just *lazy* reduction [3], but also under λ-abstraction, inside the right-hand side of an application and inside the substitution, thereby generalising previous results [19, 24, 23, 5].

As is usual with semantic interpretations, we test the correctness of our interpretation by focussing on the two main criteria:

Preservation of observations: If M terminates, then every computation of $[\![M]\!]$ (a process) signals "completion" to other processes in some manner.

Preservation of divergence: If M does not terminate, then no computation of $[\![M]\!]$ signals completion.

These two criteria are arguably the main properties that have to be shown for an interpretation, but there are many more that one could demand to hold, like preservation of *compositions*, of *reduction steps*, of *termination*, of *simulations*, of *equivalences*, etc. But, since the notions of reduction vary greatly between various calculi, it comes as no surprise that normally not all these properties are provable for any given interpretation. For example, the preservation of termination "*if M terminates, then so does $[\![M]\!]$*" is not automatically preserved.

Note that preservation of termination is not an issue when defining a semantics. For example, when representing the computable functions into the λ-calculus: to encode recursive functions a fixed-point constructor is used like $\lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$, which already on its own does not terminate, leaving interpreted functions with non-terminating parts[1]. This could be solved by enforcing a reduction strategy on the target language, typically via lazy reduction, but at the price that, for example, Church numerals no longer are a good interpretation for numbers. It should be clear that this does not imply that the λ-calculus is not a proper model of computation, so termination is not a decisive criterion on the suitability of an interpretation.

And in fact, as also stated by Clinger [11]:

> " *the formal semantics of a (...) programming language may itself be interpreted to provide an (inefficient) implementation of the language. A formal semantics need not always provide such an implementation, though, and to believe that semantics must provide an implementation leads to confusion* "

Remark that completion does not equate to termination; completion gets signalled via an observable action, like *output*, after which computation can continue, whereas termination implies that computation will eventually halt.

Milner checks precisely the two criteria above for his interpretation of the lazy λ-calculus [19] (where reductions are not allowed to take place under the λ-abstraction, nor in the right-hand side of application); Milner also considers an interpretation that respects call-by-value reduction. Sangiorgi and Walker [24] extended Milner's results also by showing that also *termination* is preserved; this is achieved mainly because each reduction step taken to reach the lazy normal form corresponds to a finite number of communication steps in the π-calculus, and in the created process

$$(\overline{\nu\vec{x}}([\![\lambda y.R]\!]^{\mathrm{M}}\, u \mid [\![\overrightarrow{x := N}]\!]^{\mathrm{M}}) \;=\; (\overline{\nu\vec{x}}(u(y).u(v).[\![R]\!]^{\mathrm{M}}\, v \mid \overrightarrow{!x(w).[\![N]\!]^{\mathrm{M}}\, w})$$

no communication is possible inside $[\![R]\!]^{\mathrm{M}}\, v$ or $[\![N]\!]^{\mathrm{M}}\, w$, since these are placed under *input*; this result comes, therefore, at the price of restricting the interpreted reduction on λ-terms to the large-step reduction to normal form of the *lazy* λ-calculus.

Milner's encoding does not respect *step-by-step* lazy β-reduction (see also [5]), but rather large-step reduction, that reduces to lazy normal form directly[2]; therefore, not all individual reduction steps are modelled. The focus of Milner and later results has been the correspondence/relation between *observable behaviour* in the lazy λ-calculus and

[1] For example, since the factorial function *fac* is defined recursively, the λ-term that represents *fac* 3 has an infinite reduction path (inside the fixed-point constructor) that does not emit any result, as well as the normal form 6.

[2] This seems to be an accepted, common fact: also Sangiorgi [23] limits reduction to lazy large-step reduction.

the π-calculus. In this setting, the problem of *full abstraction* is formulated towards an equivalence on λ-terms that is not β-equivalence, but through *applicative bisimilarity*: this has the advantage that all unsolvable terms are equated. The main result is stated as $M \approx_c^\lambda N \iff [\![M]\!] \approx_c^\pi [\![N]\!]$, which formulates that if two λ-terms have the same observable behaviour, then so do their interpretations in π, and vice-versa.

Also the interpretations we defined in the past [4, 5] do not model full reduction; this is directly caused by the fact that those interpretations place terms under *input* (the operand in an application, to be precise); the nature of the reduction relation on the π-calculus does not permit reduction under an *input*. Moreover, β-reduction is not modelled in full in [5]; rather, we showed to faithfully represent *explicit spine reduction* (as defined in that paper) that allows reduction under abstraction as well, and which normal forms correspond (*i.e.* with perhaps some substitutions pending) to β-reduction's normal forms of head reduction.

In contrast, in this paper we show that we *can* faithfully model $\lambda\mathbf{x}$'s explicit reduction in full[3] and step-by-step, into the π-calculus. Since reduction in $\lambda\mathbf{x}$ implements β-reduction, it follows that we can model β-reduction as well; the result is stated through a symmetric relation on processes, so we can model β-equality as well, which then gives that our interpretation gives, in fact, a semantics. We achieve this by generalising the logical interpretation we presented in [5].

To be precise, we define an interpretation of λ-terms $\llbracket \cdot \rrbracket^{\mathrm{F}}_{\cdot}$, for which we will show[4]:

Operational Soundness: $M \to_{\mathbf{x}} N \Rightarrow \llbracket M \rrbracket^{\mathrm{F}}_{a} \to_\pi^* \sim_{\mathrm{C}} \llbracket N \rrbracket^{\mathrm{F}}_{a}$;

Adequacy: $M =_\beta N \Rightarrow \llbracket M \rrbracket^{\mathrm{F}}_{a} \sim_{\mathrm{C}} \llbracket N \rrbracket^{\mathrm{F}}_{a}$;

Operational Completeness: $\llbracket M \rrbracket^{\mathrm{F}}_{a} \to_\pi P \Rightarrow \exists N\, [\, P \to_\pi^* \sim_{\mathrm{C}} \llbracket N \rrbracket^{\mathrm{F}}_{a} \;\&\; M \to_{\mathbf{x}}^* N]$;

Type preservation: $\Gamma \vdash_\lambda M : A \Rightarrow \llbracket M \rrbracket^{\mathrm{F}}_{a} : \Gamma \vdash_\pi a{:}A$.

1 The asynchronous π-calculus with pairing

The notion of asynchronous π-calculus that we consider in this paper is the one we also used in [4, 5], and different from other systems studied in the literature [15]. To successfully preserve assignable types, inspired by [2] we also introduce a structure over names, such that not only names but also pairs of names can be sent (but not a pair of pairs). We also introduce the *let*-construct to deal with inputs of pairs of names that get distributed over the continuation, and take the view that processes communicate by sending data over channels, so not just names, but also pairs of names.

We first define the notion of π-calculus we consider here.

Definition 1. – *Channel names* and *data* are defined by:

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

[3] Since we represent full $\lambda\mathbf{x}$-reduction, we model in particular the rewrite rule $M \to N \Rightarrow L\langle x := M\rangle \to L\langle x := N\rangle$, where reduction inside the substitution is explicitly allowed.

[4] We use $\to^+$ and $\to^*$ for the transitive, resp. reflexive and transitive closures; we use $\downarrow$ for convergence, and $\uparrow$ for divergence; $\sim_{\mathrm{C}}$ represents contextual equivalence.

– *Processes* are defined by the grammar (where x, y, z are *variables*):

$$P, Q \ ::= \ \mathbf{0} \mid P \,|\, Q \mid \,!P \mid (\nu a)\, P \mid a(x).P \mid \overline{a}\langle p\rangle \mid \mathit{let}\,\langle x, y\rangle = p \ \mathit{in}\ P$$

– A (process) context is simply a term with a hole $[\cdot]$.

– We consider n bound in $(\nu n)\, P$, x bound in $a(x).P$, and x and y to be bound in the *let*-construct. We call n free in P if it occurs in P and is not bound; we call a process *closed* if it has no free output name.

We abbreviate $a(x).\mathit{let}\,\langle y, z\rangle = x \ \mathit{in}\ P$ by $a(y, z).P$, and $(\nu m)\,(\nu n)\, P$ by $(\nu mn)\, P$, and write $\overline{a}\langle c, d\rangle$ for $\overline{a}\langle\langle c, d\rangle\rangle$. Notice that all channels are monadic.

Definition 2 (Congruence). The structural congruence relation '$\equiv$' is the smallest equivalence relation closed under contexts defined by the following rules:

$$(\nu n)\,\mathbf{0} \equiv \mathbf{0} \qquad P \,|\, \mathbf{0} \equiv P \qquad P \,|\, Q \equiv Q \,|\, P \qquad !P \equiv P \,|\, !P \qquad !P \equiv !P \,|\, !P$$
$$(P \,|\, Q) \,|\, R \equiv P \,|\, (Q \,|\, R) \qquad (\nu m)\,(\nu n)\, P \equiv (\nu n)\,(\nu m)\, P$$
$$(\nu n)\,(P \,|\, Q) \equiv P \,|\, (\nu n)\, Q, \ \mathit{if}\ n \notin \mathit{fn}(P) \qquad \mathit{let}\,\langle y, z\rangle = \langle a, b\rangle \ \mathit{in}\ R \equiv R[a/z, b/z]$$

We will consider processes modulo congruence: 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 $(P \,|\, Q) \,|\, R \equiv P \,|\, (Q \,|\, R)$, we will normally not write brackets in a parallel composition of more than two processes. We explicitly convert '*an output sent on a is to be received as input on b*' via '$a(w).\overline{b}\langle w\rangle$', which is abbreviated into $a \rightarrowtriangle b$.

Definition 3 (Reduction). 1. The *reduction relation* of the π-calculus is defined by:

$$\begin{array}{rcl} \overline{a}\langle p\rangle \,|\, a(x).Q & \rightarrow_\pi & Q[p/x] \\ P \rightarrow_\pi P' & \Rightarrow & (\nu n)\, P \rightarrow_\pi (\nu n)\, P' \\ P \rightarrow_\pi P' & \Rightarrow & P \,|\, Q \rightarrow_\pi P' \,|\, Q \\ P \equiv Q \ \&\ Q \rightarrow_\pi Q' \ \&\ Q' \equiv P' & \Rightarrow & P \rightarrow_\pi P' \end{array}$$

2. We define $\rightarrow_{\mathrm{L}\pi}$ (*lazy* reduction) as $\rightarrow_\pi$, but do not allow reduction inside closed processes[5].

Notice that $\overline{a}\langle b, c\rangle \,|\, a(x, y).Q \ \rightarrow_\pi \ Q[b/x, c/y]$.

The following notions are standard, and of use:

Definition 4. 1. We write $P \downarrow n$ (P *outputs on* n) if $P \equiv (\nu b_1 \ldots b_m)\,(\overline{n}\langle p\rangle \,|\, Q)$ for some Q, where $n \neq b_1 \ldots b_m$.

2. We write $P \Downarrow n$ (P *will output on* n) if there exists Q such that $P \rightarrow_\pi^* Q$ and $Q \downarrow n$.

3. We write $P \sqsubseteq_{\mathsf{C}} Q$ (and call $\sqsubseteq_{\mathsf{C}}$ the *contextual ordering*) if, for all contexts $\mathsf{C}[\cdot]$, and for all n, if $\mathsf{C}[P] \downarrow n$ then $\mathsf{C}[Q] \Downarrow n$.

4. We write $P \sim_{\mathrm{C}} Q$ (and call P and Q *contextually equivalent*) if and only if $P \sqsubseteq_{\mathsf{C}} Q$ and $Q \sqsubseteq_{\mathsf{C}} P$.

[5] Lazy reduction can be seen as similar to garbage collection, since it ignores processes that are silent, *i.e.* produce no visible output.

The π-calculus is equipped with a rich type theory [24]: from the basic type system for counting the arity of channels [22] to sophisticated linear types in [17], which studies a relation between Call-by-Value $\lambda\mu$ [21] and a linear π-calculus. The notion of type assignment we use here (first defined in [4]) differs from systems presented in the past in that types contain no channel information, and in that it expresses *implication*, *i.e.* has functional types and describes the '*input-output interface*' of a process.

Definition 5 (Types and Contexts). 1. Types are defined by: $A, B \ ::= \ \varphi \mid A \to B$ where φ is a basic type of which there are infinitely many.

2. A *context of inputs* Γ is a mapping from names to types, denoted as a finite set of *statements* $n{:}A$, such that the *subject* of the statements (n) are distinct. We write Γ_1, Γ_2 for the *compatible* union of Γ_1 and Γ_2 (if $n{:}A_1 \in \Gamma_1$ and $n{:}A_2 \in \Gamma_2$, then $A_1 = A_2$) , and write $\Gamma, n{:}A$ for $\Gamma, \{n{:}A\}$.
3. Contexts of *outputs* Δ, and the notions Δ_1, Δ_2 and $n{:}A, \Delta$ are defined similarly.

So, for the context $\Gamma, n{:}A$, we have $n{:}A \in \Gamma$, or Γ is not defined on n.

Definition 6 (Context assignment for π [4]). Type assignment for π-calculus is defined by the sequent system[6]:

$$(\mathbf{0}) : \frac{}{0 : \Gamma \vdash \Delta} \quad (!) : \frac{P : \Gamma \vdash \Delta}{!\,P : \Gamma \vdash \Delta} \quad (out) : \frac{}{\overline{a}\langle b\rangle : \Gamma, b{:}A \vdash a{:}A, b{:}A, \Delta}\ (a \neq b)$$

$$(\nu) :) \ \frac{P : \Gamma, a{:}A \vdash a{:}A, \Delta}{(\nu a)\,P : \Gamma \vdash \Delta} \qquad (\textit{pair-out}) : \frac{}{\overline{a}\langle b, c\rangle : \Gamma, b{:}A \vdash a{:}A \to B, c{:}B, \Delta}\ \left(\begin{smallmatrix} b \notin \Delta; \\ a, c \notin \Gamma \end{smallmatrix}\right)$$

$$(\,|\,) : \frac{P_1 : \Gamma \vdash \Delta \quad \cdots \quad P_n : \Gamma \vdash \Delta}{P_1 \,|\, \cdots \,|\, P_n : \Gamma \vdash \Delta} \quad (let) : \frac{P : \Gamma, y{:}B \vdash x{:}A, \Delta}{let\,\langle x, y\rangle = z\ in\ P : \Gamma, z{:}A \to B \vdash \Delta}\ \left(\begin{smallmatrix} y, z \notin \Delta; \\ x \notin \Gamma \end{smallmatrix}\right)$$

$$(W) : \frac{P : \Gamma \vdash \Delta}{P : \Gamma' \vdash \Delta'}\ (\Gamma' \supseteq \Gamma, \Delta' \supseteq \Delta) \qquad (in) : \frac{P : \Gamma, x{:}A \vdash x{:}A, \Delta}{a(x).P : \Gamma, a{:}A \vdash \Delta}$$

As usual, we write $P : \Gamma \vdash_{\pi} \Delta$ if there exists a derivation using these rules that has this expression in the conclusion.

Notice that the '*input-output interface of a π-process*' property is nicely preserved by all the rules; it also explains how rules show that the handling of arrow types is restricted by the type system to the rules $(\textit{let})$ and $(\textit{pair-out})$.

The above system is not trivial, since the process

$$(\nu cb)\,(x(w).\overline{c}\langle w\rangle \,|\, c(v,d).(\,!b \to v \,|\, d \to a) \,|\, !\,\ulcorner x \lrcorner\, b)$$

(the interpretation of the λ-term xx as defined below) is not typeable: the left-hand x would need the type $A \to B$, and the right-hand x the type A; x can only have *one* type, and $A \to B$ and A cannot be unified.

Example 7. The following rule is derivable:

$$(\textit{pair-in}) : \frac{P : \Gamma, y{:}B \vdash_{\pi} x{:}A, \Delta}{a(x,y).P : \Gamma, a{:}A \to B \vdash_{\pi} \Delta}\ (y, a \notin \Delta,\ x \notin \Gamma)$$

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

We leave the exploration of the logical contents of this system for future work.

The main soundness result for our notion of type assignment for π is stated as:

Theorem 8 (Witness reduction [4]). If $P : \Gamma \vdash_\pi \Delta$ and $P \to_\pi Q$, then $Q : \Gamma \vdash_\pi \Delta$.

2 The λ-calculus and λx

We assume the reader to be familiar with the λ-calculus and just repeat the definition of the relevant notions. We will look in particular at Bloo and Rose's calculus $\lambda\mathbf{x}$ [8], a version of the λ-calculus with *explicit substitution*, and show our results for $\lambda\mathbf{x}$; since $\lambda\mathbf{x}$ implements β-reduction, this implies that we also show our results for normal β-reduction (with implicit substitution).

Definition 9 (Lambda terms and β-reduction [7]).

1. The set Λ of λ-*terms* is defined by the grammar: $M, N \ ::= \ x \mid \lambda x.M \mid MN$.
2. The reduction relation $\to_\beta$ is defined by:
$$(\beta) : \ (\lambda x.M)N \ \to \ M[N/x] \qquad M \to N \ \Rightarrow \ \begin{cases} ML & \to \ NL \\ LM & \to \ LN \\ \lambda x.M & \to \ \lambda x.N \end{cases}$$
where $M[N/x]$ is the (implicit) substitution of N for x in M, which takes place immediately and silently.
3. *Lazy* reduction $\to_{\mathrm{L}}$ for the λ-calculus [3] is defined by limiting $\to_\beta$ to:
$$(\lambda x.M)N \ \to \ M[N/x] \qquad M \to N \ \Rightarrow \ ML \to NL$$
4. *Spine* reduction $\to_{\mathrm{S}}$ for the λ-calculus is defined by limiting $\to_\beta$ to:
$$(\lambda x.M)N \ \to \ M[N/x] \qquad M \to N \ \Rightarrow \ \begin{cases} ML & \to \ NL \\ \lambda x.M & \to \ \lambda x.N \end{cases}$$

We now present $\lambda\mathbf{x}$, a version of the λ-calculus with *explicit substitution*. Explicit substitution λ-calclulus treats substitution as a first-class operator, and describes all the necessary steps to effectuate a substitution. It introduces the concept of substitution within the syntax, making it *explicit*, by adding $M\langle x := N\rangle$:

Definition 10 (c.f. [8]). The syntax of the *explicit lambda calculus* $\lambda\mathbf{x}$ is defined by:
$$M, N \ ::= \ x \mid \lambda x.M \mid MN \mid M\langle x := N\rangle$$

The notion of type assignment on $\lambda\mathbf{x}$ is a natural extension of Curry's system for the λ-calculus by adding rule (cut).

Definition 11. Using the notion of types in Def. 5, type assignment for $\lambda\mathbf{x}$ is defined by:

$$(Ax) : \ \frac{}{\Gamma, x{:}A \vdash_\lambda x : A} \qquad (cut) : \ \frac{\Gamma, x{:}A \vdash_\lambda M : B \quad \Gamma \vdash_\lambda N : A}{\Gamma \vdash_\lambda M\langle x := N\rangle : B}$$

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

Explicit substitution describes explicitly the process of executing a β-reduction, where the implicit substitution of the β-reduction step is split up into reduction steps.

Definition 12. The reduction relation $\to_{\mathbf{x}}$ on terms in $\lambda\mathbf{x}$ is defined by the following rules:

$$\begin{array}{rcl} (\lambda x.M)P & \to & M\langle x := P\rangle \\ (MN)\langle x := P\rangle & \to & M\langle x := P\rangle N\langle x := P\rangle \\ (\lambda y.M)\langle x := P\rangle & \to & \lambda y.(M\langle x := P\rangle) \\ x\langle x := P\rangle & \to & P \\ y\langle x := P\rangle & \to & y, \quad y \neq x \end{array} \qquad M \to N \Rightarrow \begin{cases} ML & \to & NL \\ LM & \to & LN \\ \lambda x.M & \to & \lambda x.N \\ M\langle x := L\rangle & \to & N\langle x := L\rangle \\ L\langle x := M\rangle & \to & L\langle x := N\rangle \end{cases}$$

Since the main rules have no critical pair, reduction is confluent.

Notice that these rules allow reduction to take place also inside the substitution term, as expressed by the last rule.

The following is straightforward:

Proposition 13 ($\lambda\mathbf{x}$ implements β-reduction).
- $M \to_\beta N \Rightarrow M \to_{\mathbf{x}}^* N$.
- If M is a pure λ-term and $M \to_{\mathbf{x}} N$, then there exists L such that $N \to_{\mathbf{x}}^* L$ and $M \to_\beta^* L$.

We will show our main results for $\lambda\mathbf{x}$; since $\lambda\mathbf{x}$ implements β-reduction, we also show our results for normal β-reduction (with implicit substitution).

3 Context and background of this paper

When interpreting the λ-calculus into the π-calculus, we are faced with the need to introduce replication, and thereby possible non-termination: to encode the β-reduction rule $(\lambda x.M)N \to_\beta M[N/x]$ in the π-calculus, implicit substitution has to be modelled, for which replication has to be used. Remark that communication in the π-calculus takes place one-at-the-time, so we are forced to specify how to deal with the distributive character of implicit substitution by splitting it up into single steps, much in the spirit of explicit substitution.

Since *a priori* the required number of copies needed of N is unknown, the substitution of N for x in M has to be modelled using replication. The interpretation of $M[N/x]$ itself is the result of running the interpretation of $(\lambda x.M)N$; since no step in π introduces replication, also in the latter the interpretation of N must appear replicated. If we naively define (as Milner, interpreting under input) $[\![MN]\!]a = (\nu c)\,([\![M]\!]c \,|\, (\nu z)\,(\overline{c}\langle z\rangle.\overline{c}\langle a\rangle.\,!\,[\![N]\!]z))$, the interpretation of a redex $(\lambda x.M)N$ runs to $(\nu x)\,([\![M]\!]a \,|\, !\,[\![N]\!]x)$ (which can be seen to represent $M\langle x := N\rangle$). This now creates a termination problem. $[\![N]\!]x$ can run in infinitely many copies, each possibly exposing observable behaviour, which introduces a form of non-termination and no longer simulates the interpreted term, and since the π-calculus has no notion of *erasure*, this will always leave a non-terminating part. Moreover, $[\![N]\!]x$ is free to start running, *i.e.* runs *during* the substitution, before communication over x realises that substitution.

Milner's encoding blocks the running of $!\,[\![N]\!]x$ by placing a guard, making the synchronisation over x the deblocking action, resulting in his encoding of the λ-calculus into the (synchronous, monadic) π-calculus as defined by:

Definition 14 (Milner's encoding [19]). The *input-based encoding* of λ-terms into the *synchronous* π-calculus is defined by:

$$\begin{array}{rcll}
[\![x]\!]^{\mathrm{M}}\, a & \triangleq & \overline{x}\langle a\rangle & x \neq a \\
[\![\lambda x.M]\!]^{\mathrm{M}}\, a & \triangleq & a(x).a(b).[\![M]\!]^{\mathrm{M}}\, b & b \textit{ fresh} \\
[\![MN]\!]^{\mathrm{M}}\, a & \triangleq & (\nu c)\,([\![M]\!]^{\mathrm{M}}\, c \,|\, (\nu z)\,(\overline{c}\langle z\rangle.\overline{c}\langle a\rangle.[\![x := M]\!]^{\mathrm{M}})) & c, z \textit{ fresh} \\
[\![x := M]\!]^{\mathrm{M}} & \triangleq & !x(w).[\![M]\!]^{\mathrm{M}}\, w & w \textit{ fresh}
\end{array}$$

But blocking the encoding of operand N in an application MN from running comes at a heavy 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*. The main correctness result Milner shows is:

Theorem 15 ([19]). For closed M, either $M\uparrow$ and $[\![M]\!]^{\mathrm{M}}\, u\uparrow$, or $M \to_{\mathrm{L}} \lambda y.R\overrightarrow{[N/x]}$, and $[\![M]\!]^{\mathrm{M}}\, u \;\to_{\pi}^{*}\; (\overrightarrow{\nu x})\,([\![\lambda y.R]\!]^{\mathrm{M}}\, u \,|\, \overrightarrow{[\![x := N]\!]^{\mathrm{M}}})$.

A substantial body of research on the relationship between λ-calculus and π-calculus followed from Milner's seminal paper. *Input*-based interpretations of the λ-calculus have become the *de facto* standard, and have also been studied by Sangiorgi [23, 24], but in the context of the higher-order π-calculus, by Honda *et al.* [17] with a rich type theory, and by Thielecke [25] in the context of continuation passing style languages.

For many years it seemed that only lazy reduction could be represented. However, in [5] we presented a *logical*, *output-based* interpretation $\llbracket\cdot\rrbracket^{\mathrm{S}}$ that respects *explicit spine reduction* $\to_{\mathbf{xS}}$; it also shows that *explicit lazy reduction* $\to_{\mathbf{xL}}$ is modelled by Milner's encoding (see [5] for details).

Theorem 16 ([5]). If $M \to_{\mathbf{xL}} N$, then $[\![M]\!]^{\mathrm{M}}\, a \to_{\pi}^{*} [\![N]\!]^{\mathrm{M}}\, a$.

which gets shown for any two terms in the relation $\to_{\mathbf{xL}}$, not just for closed terms. But, in fact, for closed terms it shows the stronger result:

Corollary 17 ([5]). If M is a closed λ-term such that $M \to_{\mathbf{xL}} (\lambda y.R)\overrightarrow{\langle x := N\rangle}$, then

$$[\![M]\!]^{\mathrm{M}}\, a \;\to_{\pi}^{*}\; [\![\lambda y.R]\!]^{\mathrm{M}}\, a \,|\, \overrightarrow{[\![y := N]\!]^{\mathrm{M}}}$$

which is exactly Milner's result, but stated using *explicit* substitution, thereby avoiding the problem of how to read (the highly ambiguous) $\lambda y.R\overrightarrow{[N/x]}$.

Deviating from Milner's *input* based-encoding [19] where the abstraction $\lambda x.M$ is interpreted via an input channel x that guards the encoding of M, in [5] we introduced an interpretation that completely reverses this view: the abstraction $\lambda x.M$ gets interpreted via an asynchronous *output* which will leave the interpretation of the body M free to reduce, if possible.

The interpretation defined in [5] is:

Definition 18 (Spine interpretation [5]).

$$\begin{array}{rcl}
\llbracket x\rrbracket^{\mathrm{S}}\, a & \triangleq & x(w).\overline{a}\langle w\rangle \\
\llbracket \lambda x.M\rrbracket^{\mathrm{S}}\, a & \triangleq & (\nu x b)\,(\llbracket M\rrbracket^{\mathrm{S}}\, b \,|\, \overline{a}\langle x, b\rangle) \\
\llbracket MN\rrbracket^{\mathrm{S}}\, a & \triangleq & (\nu c)\,(\llbracket M\rrbracket^{\mathrm{S}}\, c \,|\, c(b,d).(!\llbracket N\rrbracket^{\mathrm{S}}\, b \,|\, d \rightarrow a)) \\
\llbracket M\langle x := N\rangle\rrbracket^{\mathrm{S}}\, a & \triangleq & (\nu x)\,(\llbracket M\rrbracket^{\mathrm{S}}\, a \,|\, !\llbracket N\rrbracket^{\mathrm{S}}\, x)
\end{array}$$

This interpretation is based on the relation between Natural Deduction (Intuitionistic Logic here) into the Sequent Calculus (LK) as defined by Gentzen [13], interpreting terms under *output* rather than under *input*, and using the π-calculus with pairing.

In order to make sure that the interpretation is well behaved, we make specific use of $a \twoheadrightarrow b$ [24][7], which is a process that forwards a message from channel a to b.

Although the main objective for this interpretation was to show the preservation of type assignment, we also could show that it preserves not just lazy reduction, but also the larger notion of spine reduction.

Theorem 19 ([5]). 1. If $M\uparrow$ then $\llbracket M \rrbracket^{\mathrm{S}}_{\mathrm{J}}\, a\uparrow$, and if $M \to_{\mathbf{xs}} N$, then $\llbracket M \rrbracket^{\mathrm{S}}_{\mathrm{J}}\, a \to^*_\pi \sim_{\mathrm{C}} \llbracket N \rrbracket^{\mathrm{S}}_{\mathrm{J}}\, a$.

2. If $\Gamma \vdash_\lambda M : A$, then $\llbracket M \rrbracket^{\mathrm{S}}_{\mathrm{J}}\, a : \Gamma \vdash_\pi a{:}A$.

Building on this interpretation, the interpretation $\llbracket \cdot \rrbracket^{\mathrm{F}}_{\mathrm{J}} \cdot$ we present below will improve on this result, in that we encode not only explicit spine reduction, but full explicit reduction, and thereby also full β-reduction.

4 A semantic interpretation of λ-terms for $\to_{\mathbf{x}}$ and $\to_\beta$

We will now define our full interpretation $\llbracket \cdot \rrbracket^{\mathrm{F}}_{\mathrm{J}} \cdot$ of the λ-calculus (with explicit substitution) into the π-calculus.

Our interpretation, where we name the output of terms, is defined by:

Definition 20 (Full logical interpretation).

$$\begin{array}{rcl}
\llbracket x \rrbracket^{\mathrm{F}}_{\mathrm{J}}\, a & \triangleq & x(w).\overline{a}\langle w\rangle \\
\llbracket \lambda x.M \rrbracket^{\mathrm{F}}_{\mathrm{J}}\, a & \triangleq & (\nu xb)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\mathrm{J}}\, b \,|\, \overline{a}\langle x,b\rangle) \\
\llbracket MN \rrbracket^{\mathrm{F}}_{\mathrm{J}}\, a & \triangleq & (\nu cb)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\mathrm{J}}\, c \,|\, c(v,d).(\,!b \twoheadrightarrow v \,|\, d \twoheadrightarrow a) \,|\, !\llbracket N \rrbracket^{\mathrm{F}}_{\mathrm{J}}\, b) \\
\llbracket M\langle x := N\rangle \rrbracket^{\mathrm{F}}_{\mathrm{J}}\, a & \triangleq & (\nu x)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\mathrm{J}}\, a \,|\, !\llbracket N \rrbracket^{\mathrm{F}}_{\mathrm{J}}\, x)
\end{array}$$

Notice that the main difference with respect to $\llbracket \cdot \rrbracket^{\mathrm{S}}_{\mathrm{J}} \cdot$ is the synchronisation in the interpretation of the application; where $\llbracket MN \rrbracket^{\mathrm{F}}_{\mathrm{J}} a$ contains $c(v,d).(\,!b \twoheadrightarrow v \,|\, d \twoheadrightarrow a) \,|\, !\llbracket N \rrbracket^{\mathrm{F}}_{\mathrm{J}} b$, $\llbracket MN \rrbracket^{\mathrm{S}}_{\mathrm{J}}\, a$ has $c(v,d).(\,!\llbracket N \rrbracket^{\mathrm{F}}_{\mathrm{J}}\, v \,|\, d \twoheadrightarrow a)$, where the received input name is used directly for the interpretation of N, which gets placed under *input*.

The replicated term in the interpretation of the substitution is intentionally not guarded, so can run on its own; this in contrast to the case for Milner's encoding, where it is guarded via *input*; this is intentional: we aim to interpret $\lambda\mathbf{x}$, where substitution is explicit, and reductions are allowed to take place inside N in $M\langle x := N\rangle$.

In particular: – we see a variable x as an *input* channel, and we need to retransmit its input to the output channel a that we interpret it under;

– for an abstraction $\lambda x.M$, we give the name b to the output of M; that M has input x and output b gets sent out over a – the name of $\lambda x.M$ – so that a process that wants to call on this functionality, knows which channel to send the input to, and on which channel to pick up the result;

[7] We use $a \twoheadrightarrow b$ for auxiliary forwarders generated by the interpretation, and $x(w).\overline{a}\langle w\rangle$ for the interpretation of x under the channel name a.

– for an application MN, the output of M, transmitted over c, is received as a pair $\langle v, d\rangle$ of input-output names in the *synchronisation cell* $c(v,d).(\,!b{\rightarrowtail}v \,|\, d{\rightarrowtail}a)$; the received input v name is used to redirect the output for N arriving over b (since $\llbracket N \rrbracket^{\mathrm{F}} b$ gets replicated, so does $b{\rightarrowtail}v$), enabling the simulation of substitution, and the received output name d gets redirected to the output of the application a.

– substitution is implemented via two parallel processes, which will communicate if necessary, effectuating the substitution.

– all interpretations of terms have only one free output name.

– since we aim to represent *all* reductions taking place inside an application MN, we need to express $\llbracket MN \rrbracket^{\mathrm{F}} a$ in terms of $\llbracket M \rrbracket^{\mathrm{F}} b$ and $\llbracket N \rrbracket^{\mathrm{F}} c$, where neither can appear under an *input*, since that would imply that reduction would be blocked, as is the case for the traditional approaches.

– Our interpretation generates a highly parallel implementation of λ-terms, with no nesting at all; the processes we generate are, essentially, a flat parallel composition of components like

$$x(w).\overline{a}\langle w\rangle \qquad b(v,d).(\,!a{\rightarrowtail}v \,|\, d{\rightarrowtail}c) \qquad \overline{a}\langle y, b\rangle$$

using replication where needed. Moreover, we model explicit substitution via the communication of two processes, so can faithfully take into account all steps of reduction in the explicit λ-calculus ($\lambda\mathbf{x}$ [8]) as well.

To underline the significance of our results, notice that the interpretation is not trivial, since $\lambda yz.y$ and $\lambda x.x$ are interpreted by $(\nu ybzb_1)\,(y(w).\overline{b_1}\langle w\rangle \,|\, \overline{b}\langle z, b_1\rangle \,|\, \overline{a}\langle y, b\rangle)$ and $(\nu xb)\,(x(w).\overline{b}\langle w\rangle \,|\, \overline{a}\langle x, b\rangle)$, respectively, processes that differ under $\sim_{\mathrm{C}}$.

Even though our interpretation $\llbracket \cdot \rrbracket^{\mathrm{F}} \cdot$ is fundamentally different form the one we presented in [5], we can still show that typeability is preserved[8]:

Theorem 21 ($\llbracket \cdot \rrbracket^{\mathrm{F}} \cdot$ preserves Curry types). If $\Gamma \vdash_{\lambda} M : A$, then $\llbracket M \rrbracket^{\mathrm{F}} a : \Gamma \vdash_{\pi} a{:}A$.

Example 22. The interpretation of a redex reduces as:

$$\begin{array}{ll} \llbracket (\lambda x.P)Q \rrbracket^{\mathrm{F}} a & \triangleq \\ (\nu cb)\,((\nu xb_1)\,(\llbracket P \rrbracket^{\mathrm{F}} b_1 \,|\, \overline{c}\langle x, b_1\rangle) \,|\, c(v,d).(\,!b{\rightarrowtail}v \,|\, d{\rightarrowtail}a) \,|\, !\llbracket Q \rrbracket^{\mathrm{F}} b) & \rightarrow_{\pi} \ (c) \\ (\nu bxb_1)\,(\llbracket P \rrbracket^{\mathrm{F}} b_1 \,|\, !b{\rightarrowtail}x \,|\, b_1{\rightarrowtail}a \,|\, !\llbracket Q \rrbracket^{\mathrm{F}} b) & \end{array}$$

Since reduction in $\lambda\mathbf{x}$ implements β-reduction by at least performing this step, this implies that we model each β-reduction step by at least one π-reduction.

We will now show that our interpretation fully respects the explicit reduction $\rightarrow_{\mathbf{x}}$, modulo contextual equivalence, using renaming of output. Renaming is defined and justified via the following lemma, which states that we can safely rename the output of an interpreted λ-term.

Lemma 23 (Renaming lemma). $(\nu a)\,(a{\rightarrowtail}e \,|\, \llbracket M \rrbracket^{\mathrm{F}} a) \ \sim_{\mathrm{C}} \ \llbracket M \rrbracket^{\mathrm{F}} e$.

[8] Although we are interpreting an intuitionistic system, the proof of this result strongly depends on the multi-conclusion character of $\vdash_{\pi}$.

Using this lemma, we can show that:

$$(\nu b x b_1)\,(\llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b_1 \mid !b \to x \mid b_1 \to a \mid !\llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b) \;\sim_{\mathrm{C}}\; (\nu x)\,(\llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a \mid !\llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} x)$$

which justifies the last case of Def. 20.

Since $\llbracket M\langle x := N\rangle \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a$ places $\llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a$ and $\llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} x$ in parallel, we can even show that the $\lambda\mathbf{x}$-variant of the Substitution Lemma is preserved:

Lemma 24. $\llbracket P\langle y := Q\rangle\langle x := R\rangle \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a \;\equiv\; \llbracket P\langle x := R\rangle\langle y := Q\langle x := R\rangle\rangle \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a$.

As in [19, 24], we can now show a reduction preservation result for full explicit reduction for $\lambda\mathbf{x}$, by showing that $\llbracket \cdot \rrrbracket^{\mathrm{F}}_{\mathbb{J}} \cdot$ preserves $\to_{\mathbf{x}}$ up to $\sim_{\mathrm{C}}$. As for Thm. 16, we do not require the terms to be closed:

Theorem 25. $M \to_{\mathbf{x}} N \Rightarrow \llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a \sim_{\mathrm{C}} \llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a$.

Since $\lambda\mathbf{x}$ implements β-reduction, faithful to the principle we introduced, we can show the two criteria as mentioned in the introduction:

Theorem 26 (Operational Soundness for $\to_\beta$). 1. $M \to_\beta^* N \Rightarrow \llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a \sim_{\mathrm{C}} \llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a$.

2. $M\uparrow \Rightarrow \llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a\uparrow$.

The first is shown by induction using Thm. 25; the second follows from Ex. 22.

The next example illustrates that we can now model more than lazy reduction.

Example 27. $\llbracket (\lambda x.(\lambda z.(\lambda y.M)x))N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a \qquad \triangleq, \equiv$

$$\begin{array}{ll}
(\nu c b)\,((\nu x b_1)\,(\llbracket \lambda z.(\lambda y.M)x \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b_1 \mid \overline{c}\langle x, b_1\rangle) \mid c(v,d).(!b \to v \mid d \to a) \mid !\llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b) & \to_\pi (c) \\
(\nu b)\,((\nu x b_1)\,((\nu z b_2)\,((\nu c_1 b_3)\,((\nu y b_4)\,(\llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b_4 \mid \overline{c_1}\langle y, b_4\rangle) \mid & \\
\quad c_1(v,d).(!b_3 \to v \mid d \to b_2) \mid !\llbracket x \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b_3) \mid \overline{b_1}\langle z, b_2\rangle) \mid !b \to x \mid b_1 \to a) \mid !\llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b) & \to_\pi (c_1) \\
(\nu b)\,((\nu x b_1)\,((\nu z b_2)\,((\nu b_3)\,((\nu y b_4)\,(\llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b_4 \mid !b_3 \to y \mid b_4 \to b_2) \mid !\llbracket x \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b_3) \mid & \\
\quad \overline{b_1}\langle z, b_2\rangle) \mid !b \to x \mid b_1 \to a) \mid !\llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b) & \to_\pi (b_1) \\
(\nu z b_2)\,((\nu y b_4)\,(\llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b_4 \mid b_4 \to b_2 \mid & \\
\quad (\nu b x b_3)\,(!b_3 \to y \mid !\llbracket x \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b_3 \mid !b \to x \mid !\llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b)) \mid \overline{a}\langle z, b_2\rangle) & \sim_{\mathrm{C}} (b, x, b_3) \\
(\nu z b_2)\,((\nu y b_4)\,(\llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b_4 \mid b_4 \to b_2 \mid !\llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} y) \mid \overline{a}\langle z, b_2\rangle) & \sim_{\mathrm{C}} (b_4) \\
(\nu z b_2)\,((\nu y)\,(\llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} b_2 \mid !\llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} y) \mid \overline{a}\langle z, b_2\rangle) & \triangleq \llbracket \lambda z.(M\langle y := N\rangle) \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a
\end{array}$$

Since $\sim_{\mathrm{C}}$ is symmetric, Thm. 26 immediately gives that $\llbracket \cdot \rrrbracket^{\mathrm{F}}_{\mathbb{J}} \cdot$ preserves $=_\beta$ up to $\sim_{\mathrm{C}}$, which states that our interpretation gives, in fact, a semantics for the λ-calculus:

Corollary 28 (Adequacy). If $M =_\beta N$, then $\llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a \sim_{\mathrm{C}} \llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a$.

This property gives a proof for operational completeness for $\lambda\mathbf{x}$:

Theorem 29 (Operational Completeness for $\to_\beta$ and $\to_{\mathbf{x}}$). 1. Let M be a closed term in $\lambda\mathbf{x}$. If $\llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a \to_\pi P$ then there exists $N \in \lambda\mathbf{x}$ such that $P \sim_{\mathrm{C}} \llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a$, and $M \to_{\mathbf{x}}^* N$.

2. Let M be a closed pure λ-term. If $\llbracket M \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a \to_\pi P$ then there exists $N \in \lambda$ such that $P \sim_{\mathrm{C}} \llbracket N \rrrbracket^{\mathrm{F}}_{\mathbb{J}} a$, and $M \to_{\mathbf{x}}^* N$.

Since renamings can be executed for abstractions, we can even generalise this result for lazy reduction $\to_{\mathrm{L}}$ [3] and explicit lazy reduction $\to_{\mathbf{x}\mathrm{L}}$ [5] on closed λ-terms:

Theorem 30. *Let M be closed. Then*

1. $M \to_{\mathbf{x}\mathrm{L}} \lambda y.M' \ \overrightarrow{\langle x := N \rangle}$, *then* $\llbracket M \rrbracket^{\mathrm{F}}_{\mathrm{J}} a \to^*_\pi \overrightarrow{(\nu x)}\,(\llbracket \lambda y.M' \rrbracket^{\mathrm{F}}_{\mathrm{J}} a \mid \overrightarrow{!\llbracket N \rrbracket^{\mathrm{F}}_{\mathrm{J}} x})$.
2. $M \to_{\mathrm{L}} \lambda y.M' \ \overrightarrow{[N/x]}$, *then* $\llbracket M \rrbracket^{\mathrm{F}}_{\mathrm{J}} a \to^*_\pi \overrightarrow{(\nu x)}\,(\llbracket \lambda y.M' \rrbracket^{\mathrm{F}}_{\mathrm{J}} a \mid \overrightarrow{\llbracket N \rrbracket^{\mathrm{F}}_{\mathrm{J}} x})$.

Notice that we hereby generalise Milner's result [19, 24].

As for full abstraction, we would need to show $M \approx^\lambda_c N \iff \llbracket M \rrbracket^{\mathrm{F}}_{\mathrm{J}} a \sim_{\mathrm{C}} \llbracket N \rrbracket^{\mathrm{F}}_{\mathrm{J}} a$, for some equivalence relation $\approx^\lambda_c$ on λ-terms. We have seen that $=_\beta$ is not the correct candidate for this relation, but it seems that $=_{\mathrm{BT}}$ (which equates terms that have the same Böhm tree) is suitable; we will investigate this in future work.

We cannot show "*if* $\llbracket M \rrbracket^{\mathrm{F}}_{\mathrm{J}} a \sim_{\mathrm{C}} \llbracket N \rrbracket^{\mathrm{F}}_{\mathrm{J}} a$, *then* $M =_\beta N$" (*i.e.* completeness), since different unsolvable terms like $(\lambda x.xx)(\lambda x.xx)$ and $(\lambda y.yyy)(\lambda y.yyy)$ are not β-equivalent, but are contextually equivalent; their interpretations under $\llbracket \cdot \rrbracket^{\mathrm{F}}_{\mathrm{J}} \cdot$ also never exhibit an output (see the third example in Ex. 34).

To illustrate the expressiveness of our interpretation, we now give some examples:

Example 31. First, we can run in the right-hand side of an application:

$$
\begin{array}{ll}
\llbracket I(II) \rrbracket^{\mathrm{F}}_{\mathrm{J}} a \ \triangleq\ (\nu cb)\,(\llbracket I \rrbracket^{\mathrm{F}}_{\mathrm{J}} c \mid c(v,d).(!b \to v \mid d \to a) \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\mathrm{J}} b) & \equiv, \triangleq \\
(\nu cb)\,(\llbracket I \rrbracket^{\mathrm{F}}_{\mathrm{J}} c \mid c(v,d).(!b \to v \mid d \to a) \mid (\nu c_1 b_1)\,((\nu z b_2)\,(\llbracket z \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_2 \mid \overline{c_1}\langle z, b_2 \rangle) \mid & \\
\quad c_1(v,d).(!b_1 \to v \mid d \to b) \mid !\llbracket I \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_1) \mid !\llbracket (\lambda z.z) I \rrbracket^{\mathrm{F}}_{\mathrm{J}} b) & \to_\pi (c_1) \\
(\nu cb)\,(\llbracket I \rrbracket^{\mathrm{F}}_{\mathrm{J}} c \mid c(v,d).(!b \to v \mid d \to a) \mid (\nu b_1)\,((\nu z b_2)\,(z(w).\overline{b_2}\langle w \rangle \mid !b_1 \to z \mid b_2 \to b) \mid & \\
\quad (\nu y b_4)\,(\llbracket y \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_4 \mid \overline{b_1}\langle y, b_4 \rangle) \mid !\llbracket I \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_1) \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\mathrm{J}} b) & \to_\pi (b_1, z, b_2) \\
(\nu cb)\,((\nu x b_5)\,(\llbracket x \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_5 \mid \overline{c}\langle x, b_5 \rangle) \mid c(v,d).(!b \to v \mid d \to a) \mid (\nu y b_4)\,(\llbracket y \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_4 \mid \overline{b}\langle y, b_4 \rangle) \mid & \\
\quad (\nu b_1 z)\,(!b_1 \to z \mid !\llbracket I \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_1) \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\mathrm{J}} b) & \to_\pi (c) \\
(\nu b)\,((\nu x b_5)\,(x(w).\overline{b_5}\langle w \rangle \mid !b \to x \mid b_5 \to a) \mid (\nu y b_4)\,(\llbracket y \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_4 \mid \overline{b}\langle y, b_4 \rangle) \mid & \\
\quad (\nu b_1 z)\,(!b_1 \to z \mid !\llbracket I \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_1) \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\mathrm{J}} b) & \to_\pi (b, x, b_5) \\
(\nu y b_4)\,(\llbracket y \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_4 \mid \overline{a}\langle y, b_4 \rangle) \mid (\nu b_1 z)\,(!b_1 \to z \mid !\llbracket I \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_1) \mid (\nu b x)\,(!b \to x \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\mathrm{J}} b) & \sim_{\mathrm{C}} \llbracket I \rrbracket^{\mathrm{F}}_{\mathrm{J}} a
\end{array}
$$

Notice that we had to spawn a copy of the replicated right-hand side first. So we indeed model more than just lazy or spine reduction.

The interpretation of an unsolvable term has no observable output:

$$
\begin{array}{ll}
\llbracket \Delta\Delta \rrbracket^{\mathrm{F}}_{\mathrm{J}} a \ \triangleq\ (\nu cb)\,((\nu x b_1)\,(\llbracket x x \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_1 \mid \overline{c}\langle x, b_1 \rangle) \mid c(v,d).(!b \to v \mid d \to a) \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\mathrm{J}} b) & \to_\pi (c) \\
(\nu b)\,((\nu x b_1)\,(\llbracket x x \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_1 \mid !b \to x \mid b_1 \to a) \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\mathrm{J}} b) & \equiv, \triangleq \\
(\nu b)\,((\nu x b_1)\,((\nu c_1 b_2)\,(x(w).\overline{c_1}\langle w \rangle \mid c_1(v,d).(!b_2 \to v \mid d \to b_1) \mid !\llbracket x \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_2) \mid & \\
\quad !b \to x \mid b_1 \to a) \mid (\nu y b_3)\,(\llbracket y y \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_3 \mid \overline{b}\langle y, b_3 \rangle) \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\mathrm{J}} b) & \to_\pi (b, x) \\
(\nu b)\,((\nu x b_1)\,((\nu c_1 b_2)\,((\nu y b_3)\,(\llbracket y y \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_3 \mid \overline{c_1}\langle y, b_3 \rangle) \mid & \\
\quad c_1(v,d).(!b_2 \to v \mid d \to b_1) \mid !\llbracket x \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_2) \mid !b \to x \mid b_1 \to a) \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\mathrm{J}} b) & \triangleq \\
(\nu b)\,((\nu x b_1)\,((\nu c_1 b_2)\,(\llbracket \Delta \rrbracket^{\mathrm{F}}_{\mathrm{J}} c_1 \mid & \\
\quad c_1(v,d).(!b_2 \to v \mid d \to b_1) \mid !x(w).\overline{b_2}\langle w \rangle) \mid !b \to x \mid b_1 \to a) \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\mathrm{J}} b) & \sim_{\mathrm{C}} (b, x) \\
(\nu b_1)\,((\nu c_1 b_2)\,(\llbracket \Delta \rrbracket^{\mathrm{F}}_{\mathrm{J}} c_1 \mid c_1(v,d).(!b_2 \to v \mid d \to b_1) \mid b_1 \to a \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_2) & \sim_{\mathrm{C}} (b_1) \\
(\nu c_1 b_2)\,(\llbracket \Delta \rrbracket^{\mathrm{F}}_{\mathrm{J}} c_1 \mid c_1(v,d).(!b_2 \to v \mid d \to a) \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\mathrm{J}} b_2) & \triangleq \llbracket \Delta\Delta \rrbracket^{\mathrm{F}}_{\mathrm{J}} a
\end{array}
$$

This example shows that the interpretation of $\Delta\Delta$ reduces in more than one step to an equivalent process, without exhibiting an output over a.

5 Termination for spine reduction

Since through $\llbracket \cdot \rrbracket^{\mathrm{F}}_{\cdot}$ we interpret full $\lambda\mathbf{x}$-reduction, we explicitly want the interpretation of N to be runnable in the interpretation of MN, as well as in the interpretation of the substitution term $M\langle x := N\rangle$, so we are forced to introduce potential infinite computations, whereas the corresponding λ-term might be terminating.

For the spine interpretation, this is relatively easy to fix by placing a guard on the replicated term, similar to what is done in Milner's encoding; see [6] for details. We cannot use that approach here, since it would invalidate our main representation result. We can, however, show termination for spine reduction for the λ-calculus, whilst limiting reduction for the π-calculus to lazy reduction as well, *i.e.* by limiting π's notion of reduction by not allowing reduction to take place in a term that has no visible output. This corresponds to the solution for termination of the representation of computable functions in the λ-calculus, as mentioned above.

To illustrate this, consider the interpretation of the term $M\langle x := N\rangle$ (*i.e.* the process $(\nu x)\,(\llbracket M\rrbracket^{\mathrm{S}}_{\cdot}\, a \mid !\llbracket N\rrbracket^{\mathrm{S}}_{\cdot}\, x)$), and notice that $!\llbracket N\rrbracket^{\mathrm{S}}_{\cdot}\, x \equiv \llbracket N\rrbracket^{\mathrm{S}}_{\cdot}\, x \mid \cdots \mid \llbracket N\rrbracket^{\mathrm{S}}_{\cdot}\, x \mid !\llbracket N\rrbracket^{\mathrm{S}}_{\cdot}\, x$. Now in case x appears in M, only finitely many of the $\llbracket N\rrbracket^{\mathrm{S}}_{\cdot}\, x$ will eventually communicate with the $x(w).\overline{b}\langle w\rangle$ that appear in the interpretation of M. Once these communications take place, the receiving xs will have disappeared, and we obtain a process of the shape $P \mid (\nu x)\,(!\llbracket N\rrbracket^{\mathrm{S}}_{\cdot}\, x)$; since each interpreted λ-term has *at most one* visible output (the name it is interpreted under), in this case x, which is restricted, the replicated term becomes unobservable, *i.e.* equivalent to $\mathbf{0}$.

Example 32. We can safely remove infinite computations that no longer contribute to the output:

$$
\begin{array}{lll}
\llbracket(\lambda x.I)(\Delta\Delta)\rrbracket^{\mathrm{F}}_{\cdot}\, a & \triangleq & \\
(\nu cb)\,((\nu x b_1)\,(\llbracket I\rrbracket^{\mathrm{F}}_{\cdot}\, b_1 \mid \overline{c}\langle x, b_1\rangle) \mid c(v,d).(!b\rightarrow v \mid d\rightarrow a) \mid !\llbracket\Delta\Delta\rrbracket^{\mathrm{F}}_{\cdot}\, b) & \rightarrow_{\pi} & (c) \\
(\nu b x b_1)\,(\llbracket I\rrbracket^{\mathrm{F}}_{\cdot}\, b_1 \mid !b\rightarrow x \mid b_1\rightarrow a \mid !\llbracket\Delta\Delta\rrbracket^{\mathrm{F}}_{\cdot}\, b) & \equiv & \\
(\nu b_1)\,(\llbracket I\rrbracket^{\mathrm{F}}_{\cdot}\, b_1 \mid b_1\rightarrow a) \mid (\nu b x)\,(!b\rightarrow x \mid !\llbracket\Delta\Delta\rrbracket^{\mathrm{F}}_{\cdot}\, b) & \sim_{\mathrm{C}} & \\
(\nu x)\,(\llbracket I\rrbracket^{\mathrm{F}}_{\cdot}\, a \mid !\llbracket\Delta\Delta\rrbracket^{\mathrm{F}}_{\cdot}\, x) & \equiv & \\
\llbracket I\rrbracket^{\mathrm{F}}_{\cdot}\, a \mid (\nu x)\,(!\llbracket\Delta\Delta\rrbracket^{\mathrm{F}}_{\cdot}\, x) & \sim_{\mathrm{C}} & \llbracket I\rrbracket^{\mathrm{F}}_{\cdot}\, a
\end{array}
$$

So when running the interpretation of a terminating λ-term, this results in a process where the only possible reductions take place as τ-actions, inside terms with restricted outputs, that are contextual equivalent to the process in which these parts are removed.

And in fact, we can show:

Theorem 33. *1. If $M \rightarrow^{*}_{\mathrm{S}} N$, then $\llbracket M\rrbracket^{\mathrm{F}}_{\cdot}\, a \sim_{\mathrm{C}} \llbracket N\rrbracket^{\mathrm{F}}_{\cdot}\, a$.*

2. $M\downarrow_{\mathrm{S}}$, then $\llbracket M\rrbracket^{\mathrm{F}}_{\cdot}\, a \downarrow_{\mathrm{L}\pi}$.

3. $M\uparrow_{\mathrm{S}}$, then $\llbracket M\rrbracket^{\mathrm{F}}_{\cdot}\, a \uparrow_{\mathrm{L}\pi}$.

As a counter example for full reduction, consider the following reduction:

Example 34. $\llbracket \lambda x.x(II) \rrbracket^{\mathrm{F}}_{\rfloor} a \quad \triangleq$

$$
\begin{array}{ll}
(\nu xb)\,((\nu cb_1)\,(\llbracket x \rrbracket^{\mathrm{F}}_{\rfloor} c \,|\, c(v,d).(!b_1 \rightarrow v \,|\, d \rightarrow b) \,|\, !\llbracket II \rrbracket^{\mathrm{F}}_{\rfloor} b_1) \,|\, \overline{a}\langle x,b\rangle) & \equiv, \triangleq \\
(\nu xb)\,((\nu cb_1)\,(\llbracket x \rrbracket^{\mathrm{F}}_{\rfloor} c \,|\, c(v,d).(!b_1 \rightarrow v \,|\, d \rightarrow b) \,|\, (\nu c_1 b_2)\,((\nu z b_3)\,(\llbracket z \rrbracket^{\mathrm{F}}_{\rfloor} b_3 \,|\, \overline{c_1}\langle z,b_3\rangle) \,| & \\
\quad c_1(v,d).(!b_2 \rightarrow v \,|\, d \rightarrow b_1) \,|\, !\llbracket I \rrbracket^{\mathrm{F}}_{\rfloor} b_2) \,|\, !\llbracket (\lambda z.z) I \rrbracket^{\mathrm{F}}_{\rfloor} b_1) \,|\, \overline{a}\langle x,b\rangle) & \rightarrow_\pi \quad (c_1) \\
(\nu xb)\,((\nu cb_1)\,(\llbracket x \rrbracket^{\mathrm{F}}_{\rfloor} c \,|\, c(v,d).(!b_1 \rightarrow v \,|\, d \rightarrow b) \,|\, (\nu b_2)\,((\nu z b_3)\,(z(w).\overline{b_3}\langle w\rangle \,|\, !b_2 \rightarrow z \,|\, b_3 \rightarrow b_1) \,| & \\
\quad !(\nu y b_5)\,(\llbracket y \rrbracket^{\mathrm{F}}_{\rfloor} b_5 \,|\, \overline{b_2}\langle y,b_5\rangle))) \,|\, !\llbracket II \rrbracket^{\mathrm{F}}_{\rfloor} b_1) \,|\, \overline{a}\langle x,b\rangle) & \rightarrow_\pi \quad (b_2,z,b_3) \\
(\nu xb)\,((\nu cb_1)\,(\llbracket x \rrbracket^{\mathrm{F}}_{\rfloor} c \,|\, c(v,d).(!b_1 \rightarrow v \,|\, d \rightarrow a) \,|\, \llbracket I \rrbracket^{\mathrm{F}}_{\rfloor} b_1 \,| & \\
\quad (\nu b_2 z)\,(!b_2 \rightarrow z \,|\, !\llbracket I \rrbracket^{\mathrm{F}}_{\rfloor} b_2) \,|\, !\llbracket II \rrbracket^{\mathrm{F}}_{\rfloor} b_1) \,|\, \overline{a}\langle x,b\rangle) & \sim_{\mathrm{C}} \\
(\nu xb)\,((\nu cb_1)\,(\llbracket x \rrbracket^{\mathrm{F}}_{\rfloor} c \,|\, c(v,d).(!b_1 \rightarrow v \,|\, d \rightarrow a) \,|\, \llbracket I \rrbracket^{\mathrm{F}}_{\rfloor} b_1 \,|\, !\llbracket II \rrbracket^{\mathrm{F}}_{\rfloor} b_1) \,|\, \overline{a}\langle x,b\rangle) & \sim_{\mathrm{C}} \quad (\cdots) \\
(\nu xb)\,((\nu cb_1)\,(\llbracket x \rrbracket^{\mathrm{F}}_{\rfloor} c \,|\, c(v,d).(!b_1 \rightarrow v \,|\, d \rightarrow a) \,|\, !\llbracket I \rrbracket^{\mathrm{F}}_{\rfloor} b_1) \,|\, \overline{a}\langle x,b\rangle) & \triangleq \llbracket \lambda x.xI \rrbracket^{\mathrm{F}}_{\rfloor} a
\end{array}
$$

Notice that we have reduced here under a bound name b_1, so the reduction in π is not lazy, and the reduction $\lambda x.x(II) \rightarrow \lambda x.xI$ is not a spine reduction.

Moreover, the notion of type assignment we defined above catches this: since $\llbracket N \rrbracket^{\mathrm{F}}_{\rfloor} x$ has at most only one visible output (which is x), the process $(\nu x)\,(!\llbracket N \rrbracket^{\mathrm{F}}_{\rfloor} x)$ has *no* visible output, and can therefore be typed by $(\nu x)\,(!\llbracket N \rrbracket^{\mathrm{F}}_{\rfloor} x) : \Gamma \vdash_\pi$, *i.e.* with an empty right-hand context; this means that the type system is capable of indicating those processes that can be ignored and stopped from running, which could be used for a type-based reduction. We leave this for future research.

Conclusions and Future Work

Our research focusses on the expressive power of the π-calculus, which has proven to provide a conceptual abstraction to understand passing private information and modelling mobile systems. Moreover, it has been shown that in principle (limited) functional and objective programming languages could be developed [24]. With the body of our work, we have also shown that the π-calculus can give computational content to the implicative fragment of the Intuitionistic Logic through its related calculus, where cut-elimination corresponds to a computational step, without restrictions.

We have found a new, simple and intuitive interpretation of λ-terms in π that respects explicit reduction, and encompasses Milner's lazy reduction on closed terms. We have shown that, for our context assignment system that uses the type constructor $\rightarrow$ for π and is based on classical logic, typeable λ-terms are interpreted by our interpretation as typeable π-processes, preserving the types.

By the nature of our interpretation, where all λ-terms are interpreted as a flat parallel composition of $x(w).\overline{a}\langle w\rangle$, $b(v,d).(!a \rightarrow v \,|\, d \rightarrow c)$, and $\overline{a}\langle y,b\rangle$, it is clear that the interpretations of a variable x in a term M occur in parallel as $x(w).\overline{a_1}\langle w\rangle \,|\, \cdots \,|\, x(w).\overline{a_n}\langle w\rangle$ inside $\llbracket M \rrbracket^{\mathrm{F}}_{\rfloor} a$; it seems plausible to use *broadcast communication* [14] – this would also solve the problem of termination. We will investigate this in future work.

We will also look on how to extend our results to a full equivalence relation on the λ-calculus that interprets λ-terms by their Böhm tree; we expect to be able to show a full abstraction result with respect to this relation.

The next naturally arising issue is to understand what the interpretations we presented here tell us about either the interpreted calculi, or the π-calculus itself. For one, we need to focus on the notion of type assignment we have used here, and study it in its own right, especially its relation with logic.

References

1. M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *JFP*, 1(4):375-416, 1991.
2. M. Abadi and A. Gordon. A calculus for cryptographic protocols: The spi calculus. *CCS'97*, pp 36-47, 1997.
3. S. Abramsky. The lazy lambda calculus. In *Research topics in functional programming*, pp 65-116. Addison-Wesley, 1990.
4. S. van Bakel, L. Cardelli, and M.G. Vigliotti. From $\mathcal{X}$ to π; Representing the Classical Sequent Calculus in the π-calculus. *CL&C'08*, 2008.
5. S. van Bakel and M.G. Vigliotti. A logical interpretation of the λ-calculus into the π-calculus, preserving spine reduction and types. *CONCUR'09*, LNCS 5710, pp 84-98, 2009.
6. S. van Bakel and M.G. Vigliotti. Implicative Logic based encoding of the λ-calculus into the π-calculus. Submitted.
7. H. Barendregt. *The Lambda Calculus: its Syntax and Semantics*. North-Holland, 1984.
8. R. Bloo and K.H. Rose. Preservation of Strong Normalisation in Named Lambda Calculi with Explicit Substitution and Garbage Collection. *CSN'95*, pp 62-72, 1995.
9. A. Church. A note on the entscheidungsproblem. *JSL*, 1(1):40-41, 1936.
10. M. Cimini, C. Sacerdoti Coen, and D. Sangiorgi. $\overline{\lambda}\mu\tilde{\mu}$ calculus, π-calculus, and abstract machines. *EXPRESS'09*, 2009.
11. M.D. Clinger. *Foundations of Actor Semantics*. MIT Artificial Intelligence Laboratory, 1981. Technical Report 633.
12. H.B. Curry. Grundlagen der Kombinatorischen Logik. *American Journal of Mathematics*, 52:509–536, 789–834, 1930.
13. G. Gentzen. Untersuchungen über das Logische Schliessen. *Mathematische Zeitschrift*, 39:176-210 and 405-431, 1935.
14. M. Hennessy and J. Rathke. Bisimulations for a Calculus of Broadcasting Systems. *TCS*, 200(1-2):225-260, 1998.
15. K. Honda and M. Tokoro. An object calculus for asynchronous communication. *ECOOP'91*, LNCS 512, pp 133-147, 1991.
16. K. Honda and N. Yoshida. On the reduction-based process semantics. *TCS*, 151:437-486, 1995.
17. K. Honda, N. Yoshida, and M. Berger. Control in the π-Calculus. *CW'04*, 2004.
18. S. Maffeis and I.C.C. Phillips. On the Computational Strength of Pure Ambient Calculi. *Express'03*, ENTCS 91, 2004.
19. R. Milner. Functions as processes. *MSCS*, 2(2):269-310, 1992.
20. U. Nestmann. What Is a Good Encoding of Guarded Choice? *EXPRESS'97*, ENTCS 7, pp 206-226, 1997.
21. M. Parigot. An algorithmic interpretation of classical natural deduction. *LPAR'92*, LNCS 624, pp 190-201, 1992.
22. B.C. Pierce and D. Sangiorgi. Typing and Subtyping for Mobile Processes. *MSCS*, 6(5):409-453, 1996.
23. D. Sangiorgi. Lazy functions and mobile processes. Rapport de Recherche 2515, INRIA, Sophia-Antipolis, France, 1995.
24. D. Sangiorgi and D. Walker. *The Pi-Calculus*. Cambridge University Press, 2001.
25. H. Thielecke. *Categorical Structure of Continuation Passing Style*. PhD thesis, University of Edinburgh, 1997. LFCS technical report ECS-LFCS-97-376.
26. M.G. Vigliotti. *Reduction Semantics for Ambient Calculi*. PhD thesis, Imperial College London, 2004.

A Proof of the main results

Theorem 21 *If* $\Gamma \vdash_\lambda M : A$, *then* $\llbracket M \rrbracket^{\mathrm{F}} a : \Gamma \vdash_\pi a{:}A$.

PROOF. By induction on the structure of derivations in $\vdash_\lambda$; notice that we use implicit weakening.

(Ax) : Then $M = x$, and $\Gamma = \Gamma', x{:}A$. Notice that $\llbracket x \rrbracket^{\mathrm{F}} a = x(w).\overline{a}\langle w\rangle$, and that

$$\dfrac{\dfrac{}{\overline{a}\langle w\rangle : \Gamma', w{:}A \vdash_\pi a{:}A, w{:}A}\,(out)}{x(w).\overline{a}\langle w\rangle : \Gamma', x{:}A \vdash_\pi a{:}A}\,(in)$$

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

$$\dfrac{\dfrac{\dfrac{\boxed{\mathcal{D}}\qquad \llbracket N \rrbracket^{\mathrm{F}} b : \Gamma, x{:}C \vdash_\pi b{:}D \qquad \dfrac{}{\overline{a}\langle x, b\rangle : x{:}C \vdash_\pi a{:}C{\to}D, b{:}D}\,(pair\text{-}out)}{\llbracket N \rrbracket^{\mathrm{F}} b \,|\, \overline{a}\langle x, b\rangle : x{:}C \vdash_\pi a{:}C{\to}D, b{:}D}\,(|)}{(\nu b)\,(\llbracket N \rrbracket^{\mathrm{F}} b \,|\, \overline{a}\langle x, b\rangle) : \Gamma, x{:}C \vdash_\pi a{:}C{\to}D}\,(\nu)}{(\nu xb)\,(\llbracket N \rrbracket^{\mathrm{F}} b \,|\, \overline{a}\langle x, b\rangle) : \Gamma \vdash_\pi a{:}C{\to}D}\,(\nu)$$

$(\to E)$: Then $M = PQ$, and there exists B such that $\Gamma \vdash_\lambda P : B{\to}A$ and $\Gamma \vdash_\lambda Q : B$, and $\llbracket PQ \rrbracket^{\mathrm{F}} a = (\nu cb)\,(\llbracket P \rrbracket^{\mathrm{F}} c \,|\, c(v,d).(!b{\to}v \,|\, d{\to}a) \,|\, !\llbracket Q \rrbracket^{\mathrm{F}} b)$. By induction, there exist $\mathcal{D}_1 :: \llbracket P \rrbracket^{\mathrm{F}} c : \Gamma \vdash_\pi c{:}B{\to}A$ and $\mathcal{D}_2 :: \llbracket Q \rrbracket^{\mathrm{F}} b : \Gamma \vdash_\pi b{:}B$, and we can construct:

$$\dfrac{\dfrac{\dfrac{\dfrac{\boxed{\mathcal{D}_1}}{\llbracket P \rrbracket^{\mathrm{F}} c : \Gamma \vdash_\pi c{:}B{\to}A}}{!\llbracket P \rrbracket^{\mathrm{F}} c : \Gamma \vdash_\pi c{:}B{\to}A}\,(!) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{v}\langle w\rangle : \Gamma, w{:}B \vdash_\pi v{:}B, w{:}B, \Delta}\,(out)}{b{\to}v : b{:}B \vdash_\pi v{:}B}\,(in)}{!b{\to}v : b{:}B \vdash_\pi v{:}B}\,(!) \qquad \dfrac{\dfrac{}{\overline{a}\langle w\rangle : \Gamma, w{:}A \vdash_\pi a{:}A, w{:}A, \Delta}\,(out)}{d{\to}a : d{:}A \vdash_\pi a{:}A}\,(in)}{!b{\to}v \,|\, d{\to}a : \Gamma, b{:}B, d{:}A \vdash_\pi v{:}B, a{:}A}\,(|)}{c(v,d).(!b{\to}v \,|\, d{\to}a) : \Gamma, b{:}B, c{:}B{\to}A \vdash_\pi a{:}A}}{\vdots} \qquad \dfrac{\boxed{\mathcal{D}_2}}{\llbracket Q \rrbracket^{\mathrm{F}} b : \Gamma \vdash_\pi b{:}B}}{!\llbracket Q \rrbracket^{\mathrm{F}} b : \Gamma \vdash_\pi b{:}B}\,(!)}{!\llbracket P \rrbracket^{\mathrm{F}} c \,|\, c(v,d).(!b{\to}v \,|\, d{\to}a) \,|\, !\llbracket Q \rrbracket^{\mathrm{F}} b : \Gamma, b{:}B, c{:}B{\to}A \vdash_\pi c{:}B{\to}A, a{:}A}\,(|)}{(\nu b)\,(!\llbracket P \rrbracket^{\mathrm{F}} c \,|\, c(v,d).(!b{\to}v \,|\, d{\to}a) \,|\, !\llbracket Q \rrbracket^{\mathrm{F}} b) : \Gamma, c{:}B{\to}A \vdash_\pi c{:}B{\to}A, a{:}A}\,(\nu)}{(\nu cb)\,(\llbracket P \rrbracket^{\mathrm{F}} c \,|\, c(v,d).(!b{\to}v \,|\, d{\to}a) \,|\, !\llbracket Q \rrbracket^{\mathrm{F}} b) : \Gamma \vdash_\pi a{:}A}\,(\nu)$$

(cut) : Then $M = P\langle x := Q\rangle$, and $\Gamma, x{:}B \vdash_\lambda P : A$ and $\Gamma \vdash_\lambda Q : B$ for some B, and, by definition, $\llbracket P\langle x := Q\rangle \rrbracket^{\mathrm{F}} a = (\nu x)\,(\llbracket P \rrbracket^{\mathrm{F}} a \,|\, !\llbracket Q \rrbracket^{\mathrm{F}} x)$. By induction, there exist

$\mathcal{D}_1 :: \llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} c : \Gamma, x{:}B \vdash_{\pi} a{:}A$ and $\mathcal{D}_2 :: \llbracket Q \rrbracket^{\mathrm{F}}_{\mathbb{J}} x : \Gamma \vdash_{\pi} x{:}B$, and we can construct:

$$\dfrac{\dfrac{\begin{array}{c}\boxed{\mathcal{D}_1}\\ \llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} a : \Gamma, x{:}B \vdash_{\pi} a{:}A\end{array} \quad \dfrac{\begin{array}{c}\boxed{\mathcal{D}_2}\\ \llbracket Q \rrbracket^{\mathrm{F}}_{\mathbb{J}} x : \Gamma \vdash_{\pi} x{:}B\end{array}}{!\llbracket Q \rrbracket^{\mathrm{F}}_{\mathbb{J}} x : \Gamma \vdash_{\pi} x{:}B}\,(!)}{\llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} a \,\big|\, !\llbracket Q \rrbracket^{\mathrm{F}}_{\mathbb{J}} x : \Gamma, x{:}B \vdash_{\pi} x{:}B, a{:}A}\,(|)}{(\nu x)\,(\llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} a \,\big|\, !\llbracket Q \rrbracket^{\mathrm{F}}_{\mathbb{J}} x) : \Gamma \vdash_{\pi} a{:}A}\,(\nu)$$

□

The following lemmas state some basic properties on processes that are relevant to our results.

Lemma 35. *The following are admissible.*

1. $!P \equiv !P \,\big|\, !P$.
2. $(\nu a)\,((\nu b)\,(Q \,\big|\, R) \,\big|\, !P) \;\equiv\; (\nu b)\,((\nu a)\,(Q \,\big|\, !P) \,\big|\, (\nu a)\,(R \,\big|\, !P))$ *provided a is the name of the only (private) channel that is used between Q and P, and between R and P (in one direction only).*
3. $(\nu x)\,(!\llbracket N \rrbracket^{\mathrm{F}}_{\mathbb{J}} b \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} x) \sim_{\mathrm{C}} !((\nu x)\,\llbracket N \rrbracket^{\mathrm{F}}_{\mathbb{J}} b \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} x)$.

PROOF. Easy. □

Lemma 24 $\llbracket P\,\langle y := Q\rangle\,\langle x := R\rangle \rrbracket^{\mathrm{F}}_{\mathbb{J}} a \;\equiv\; \llbracket P\,\langle x := R\rangle\,\langle y := Q\,\langle x := R\rangle\rangle \rrbracket^{\mathrm{F}}_{\mathbb{J}} a$.

PROOF.

$$\begin{array}{lll}
\llbracket P\,\langle y := Q\rangle\,\langle x := R\rangle \rrbracket^{\mathrm{F}}_{\mathbb{J}} a & \triangleq & \\
(\nu x)\,((\nu y)\,(\llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} a \,\big|\, !\llbracket Q \rrbracket^{\mathrm{F}}_{\mathbb{J}} y) \,\big|\, !\llbracket R \rrbracket^{\mathrm{F}}_{\mathbb{J}} x) & \equiv & \\
(\nu xy)\,(\llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} a \,\big|\, !\llbracket Q \rrbracket^{\mathrm{F}}_{\mathbb{J}} y \,\big|\, !\llbracket R \rrbracket^{\mathrm{F}}_{\mathbb{J}} x) & \equiv & (35(1)) \\
(\nu xy)\,(\llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} a \,\big|\, !\llbracket R \rrbracket^{\mathrm{F}}_{\mathbb{J}} x \,\big|\, !\llbracket Q \rrbracket^{\mathrm{F}}_{\mathbb{J}} y \,\big|\, !\llbracket R \rrbracket^{\mathrm{F}}_{\mathbb{J}} x) & \equiv & \\
(\nu y)\,((\nu x)\,(\llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} a \,\big|\, !\llbracket R \rrbracket^{\mathrm{F}}_{\mathbb{J}} x) \,\big|\, (\nu x)\,(!\llbracket Q \rrbracket^{\mathrm{F}}_{\mathbb{J}} y \,\big|\, !\llbracket R \rrbracket^{\mathrm{F}}_{\mathbb{J}} x)) & \equiv & (35(3)) \\
(\nu y)\,((\nu x)\,(\llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} a \,\big|\, !\llbracket R \rrbracket^{\mathrm{F}}_{\mathbb{J}} x) \,\big|\, !(\nu x)\,(\llbracket Q \rrbracket^{\mathrm{F}}_{\mathbb{J}} y \,\big|\, !\llbracket R \rrbracket^{\mathrm{F}}_{\mathbb{J}} x)) & \equiv & \\
(\nu y)\,((\nu x)\,(\llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} a \,\big|\, !\llbracket R \rrbracket^{\mathrm{F}}_{\mathbb{J}} x) \,\big|\, !\llbracket Q\,\langle x := R\rangle \rrbracket^{\mathrm{F}}_{\mathbb{J}} y) & \triangleq & \\
(\nu y)\,(\llbracket P\langle x := R\rangle \rrbracket^{\mathrm{F}}_{\mathbb{J}} a \,\big|\, !\llbracket (Q\langle x := R\rangle) \rrbracket^{\mathrm{F}}_{\mathbb{J}} y) & \triangleq & \\
\llbracket P\,\langle x := R\rangle\,\langle y := Q\,\langle x := R\rangle\rangle \rrbracket^{\mathrm{F}}_{\mathbb{J}} a & &
\end{array}$$

□

Theorem 25 $M \rightarrow_{\mathbf{x}} N \Rightarrow \llbracket M \rrbracket^{\mathrm{F}}_{\mathbb{J}} a \sim_{\mathrm{C}} \llbracket N \rrbracket^{\mathrm{F}}_{\mathbb{J}} a$.

PROOF. By induction on explicit reduction.

$$\begin{array}{lll}
(\lambda x.M)P \rightarrow M\langle x := P\rangle : \llbracket (\lambda x.M)P \rrbracket^{\mathrm{F}}_{\mathbb{J}} a & \triangleq & \\
\quad (\nu cb)\,((\nu xb_1)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\mathbb{J}} b_1 \,\big|\, \overline{c}\langle x, b_1\rangle) \,\big|\, c(v,d).(!b{\rightarrow}v \,\big|\, d{\rightarrow}a) \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} b) & \sim_{\mathrm{C}} & (c) \\
\quad (\nu cb)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\mathbb{J}} b_1 \,\big|\, !b{\rightarrow}x \,\big|\, b_1{\rightarrow}a \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} b) & \sim_{\mathrm{C}} & (\nu x)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\mathbb{J}} a \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}}_{\mathbb{J}} x)
\end{array}$$

$$
\begin{array}{ll}
(MN)\langle x := P\rangle \to M\langle x := P\rangle N\langle x := P\rangle : \llbracket MN\,\langle x := P\rangle \rrbracket^{\mathrm{F}} a & \triangleq \\
\quad (\nu x)\,(\llbracket MN \rrbracket^{\mathrm{F}} a \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}} x) & \triangleq \\
\quad (\nu x)\,((\nu cb)\,(\llbracket M \rrbracket^{\mathrm{F}} c \,\big|\, c(v,d).(!b{\to}v \,\big|\, d{\to}a) \,\big|\, !\llbracket N \rrbracket^{\mathrm{F}} b) \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}} x) & \equiv \ (35(2)) \\
\quad (\nu cb)\,((\nu x)\,(\llbracket M \rrbracket^{\mathrm{F}} c \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}} x) \,\big|\, c(v,d).(!b{\to}v \,\big|\, d{\to}a) \,\big|\, (\nu x)\,(!\llbracket N \rrbracket^{\mathrm{F}} b \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}} x)) & \\
& \equiv \ (35(3)) \\
\quad (\nu cb)\,((\nu x)\,(\llbracket M \rrbracket^{\mathrm{F}} c \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}} x) \,\big|\, c(v,d).(!b{\to}v \,\big|\, d{\to}a) \,\big|\, !\llbracket N\,\langle x := P\rangle \rrbracket^{\mathrm{F}} b) & \triangleq \\
\quad (\nu cb)\,(\llbracket M\langle x := P\rangle \rrbracket^{\mathrm{F}} c \,\big|\, c(v,d).(!b{\to}v \,\big|\, d{\to}a) \,\big|\, !\llbracket N\langle x := P\rangle \rrbracket^{\mathrm{F}} b) & \triangleq \\
\quad \llbracket M\,\langle x := P\rangle N\,\langle x := P\rangle \rrbracket^{\mathrm{F}} a &
\end{array}
$$

$$
\begin{array}{ll}
(\lambda\langle y.M)x := P\rangle \to \lambda y.(M\langle x := P\rangle) : \llbracket (\lambda y.M)\,\langle x := P\rangle \rrbracket^{\mathrm{F}} a & \triangleq \\
\quad (\nu x)\,((\nu yb)\,(\llbracket M \rrbracket^{\mathrm{F}} b \,\big|\, \overline{a}\langle y,b\rangle) \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}} x) & \equiv \\
\quad (\nu yb)\,((\nu x)\,(\llbracket M \rrbracket^{\mathrm{F}} b \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}} x) \,\big|\, \overline{a}\langle y,b\rangle) & \triangleq \ \llbracket \lambda y.M\,\langle x := P\rangle \rrbracket^{\mathrm{F}} a
\end{array}
$$

$$
\begin{array}{llll}
x\langle x := P\rangle \to P : \llbracket x\,\langle x := P\rangle \rrbracket^{\mathrm{F}} a & \triangleq & (\nu x)\,(\llbracket x \rrbracket^{\mathrm{F}} a \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}} x) & \equiv \\
\quad (\nu x)\,(x(w).\overline{a}\langle w\rangle \,\big|\, \llbracket P \rrbracket^{\mathrm{F}} x \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}} x) & \sim_{\mathrm{C}} (23) & (\nu x)\,(\llbracket P \rrbracket^{\mathrm{F}} a \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}} x) & \equiv \\
\quad \llbracket P \rrbracket^{\mathrm{F}} a \,\big|\, (\nu x)\,(!\llbracket P \rrbracket^{\mathrm{F}} x) & \sim_{\mathrm{C}} & \llbracket P \rrbracket^{\mathrm{F}} a &
\end{array}
$$

$$
\begin{array}{ll}
y\langle x := P\rangle \to y,\ y \neq x : & \llbracket y\,\langle x := P\rangle \rrbracket^{\mathrm{F}} a \ \triangleq \ (\nu x)\,(\llbracket y \rrbracket^{\mathrm{F}} a \,\big|\, !\llbracket P \rrbracket^{\mathrm{F}} x) \ \equiv \\
\quad \llbracket y \rrbracket^{\mathrm{F}} a \,\big|\, (\nu x)\,(!\llbracket P \rrbracket^{\mathrm{F}} x) & \sim_{\mathrm{C}} \ \llbracket y \rrbracket^{\mathrm{F}} a
\end{array}
$$

$$
\begin{array}{ll}
M \to N \Rightarrow ML \to NL : \llbracket ML \rrbracket^{\mathrm{F}} a & \triangleq \\
\quad (\nu cb)\,(\llbracket M \rrbracket^{\mathrm{F}} c \,\big|\, c(v,d).(!b{\to}v \,\big|\, d{\to}a) \,\big|\, !\llbracket L \rrbracket^{\mathrm{F}} b) & \sim_{\mathrm{C}} \ (IH) \\
\quad (\nu cb)\,(\llbracket N \rrbracket^{\mathrm{F}} c \,\big|\, c(v,d).(!b{\to}v \,\big|\, d{\to}a) \,\big|\, !\llbracket L \rrbracket^{\mathrm{F}} b) & \triangleq \ \llbracket NL \rrbracket^{\mathrm{F}} a
\end{array}
$$

$$
\begin{array}{ll}
M \to N \Rightarrow LM \to LN : \llbracket LM \rrbracket^{\mathrm{F}} a & \triangleq \\
\quad (\nu cb)\,(\llbracket L \rrbracket^{\mathrm{F}} c \,\big|\, c(v,d).(!b{\to}v \,\big|\, d{\to}a) \,\big|\, !\llbracket M \rrbracket^{\mathrm{F}} b) & \sim_{\mathrm{C}} \ (IH) \\
\quad (\nu cb)\,(\llbracket L \rrbracket^{\mathrm{F}} c \,\big|\, c(v,d).(!b{\to}v \,\big|\, d{\to}a) \,\big|\, !\llbracket N \rrbracket^{\mathrm{F}} b) & \triangleq \ \llbracket LN \rrbracket^{\mathrm{F}} a
\end{array}
$$

$$
\begin{array}{ll}
M \to N \Rightarrow \lambda x.M \to \lambda x.N : & \llbracket \lambda x.M \rrbracket^{\mathrm{F}} a \ \triangleq \ (\nu xb)\,(\llbracket M \rrbracket^{\mathrm{F}} b \,\big|\, \overline{a}\langle x,b\rangle) \ \sim_{\mathrm{C}} (IH) \\
\quad (\nu xb)\,(\llbracket N \rrbracket^{\mathrm{F}} b \,\big|\, \overline{a}\langle x,b\rangle) \triangleq & \llbracket \lambda x.N \rrbracket^{\mathrm{F}} a
\end{array}
$$

$$
\begin{array}{llll}
M \to N \Rightarrow M\langle x := L\rangle \to N\langle x := L\rangle : & \llbracket M\,\langle x := L\rangle \rrbracket^{\mathrm{F}} a & \triangleq & \\
\quad (\nu x)\,(\llbracket M \rrbracket^{\mathrm{F}} a \,\big|\, !\llbracket L \rrbracket^{\mathrm{F}} x) \ \sim_{\mathrm{C}} (IH) & (\nu x)\,(\llbracket N \rrbracket^{\mathrm{F}} a \,\big|\, !\llbracket L \rrbracket^{\mathrm{F}} x) & \triangleq & \llbracket N\,\langle x := L\rangle \rrbracket^{\mathrm{F}} a
\end{array}
$$

$$
\begin{array}{llll}
M \to N \Rightarrow L\langle x := M\rangle \to L\langle x := N\rangle : & \llbracket L\,\langle x := M\rangle \rrbracket^{\mathrm{F}} a & \triangleq & \\
\quad (\nu x)\,(\llbracket L \rrbracket^{\mathrm{F}} a \,\big|\, !\llbracket M \rrbracket^{\mathrm{F}} x) \ \sim_{\mathrm{C}} (IH) & (\nu x)\,(\llbracket L \rrbracket^{\mathrm{F}} a \,\big|\, !\llbracket N \rrbracket^{\mathrm{F}} x) & \triangleq & \llbracket L\,\langle x := N\rangle \rrbracket^{\mathrm{F}} a \ \square
\end{array}
$$