An output-based semantics 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 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 reduction with respect to a notion of 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 [23, 27, 26, 7] and typed [21] λ-calculus [13, 8] and object oriented calculi [19, 27] have been shown. Also, various encodings of calculi that represent classical logic have been recently proposed [21, 5, 14].

In this paper we investigate the expressive power of the asynchronous π-calculus [23] extended with pairing by showing a new compositional semantic interpretation of $\lambda\mathbf{x}$, the explicit substitution λ-calculus [1, 10] that fully respects *each individual reduction step i.e.* also under abstraction, in the right-hand side of an application, and inside the explicit substitution; through this result, using the fact that explicit substitution implements implicit substitution, we will show that our interpretation gives a semantics [1]for the λ-calculus.

The advantage of considering explicit substitution rather than the standard implicit substitution of the λ-calculusas considered in [27] was already argued in [7]. In that work 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. 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 [23, 27, 26, 7].

As is usual with semantic interpretations, we test the correctness of our interpretation by focussing on the two main criteria (see [22]), which offer a guarantee that reduction will be simulated correctly:

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

[1] Note that we do not present an *implementation* of the λ-calculus.

(*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.

Take the λ-calculus and Combinatory Logic (CL) [16]; although these are naturally linked, the fact that reduction is strong in the first and weak in the second makes preservation of termination dubious. Although the encoding of the λ-calculus into CL is well behaved, the reverse encoding does preserves termination:

i) $t = \mathbf{S}(\mathbf{K}(\mathbf{SII}))(\mathbf{K}(\mathbf{SII}))$ is a normal form, but $[\![t]\!]_\lambda \to_\beta^* +\lambda c.(\lambda x.xx)(\lambda x.xx)$, which does not have a β-normal form, not even a head-normal form, so no observable behaviour.

ii) $t = \mathbf{SK}((\mathbf{SII})(\mathbf{SII}))$ has no normal form, while $[\![t]\!]_\lambda \to_\beta^* +\lambda x.x$.

When mapping one kind of reduction onto another, not all properties are preserved.

(See [6] for a discussion on approximation semantics and full abstraction for Combinatory Systems).

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[2]. 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: termination is not a decisive criterion on the suitability of an interpretation.

And in fact, as also stated by Clinger [15]:

> "*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.

In general, in the concurrency setting there are many criteria used to established whether an interpretation is good; here we focus on the previous two criteria, that will be formally translated into:

- Preservation of execution steps: if $M \to N$ then $[\![M]\!] \to^* \simeq [\![N]\!]$;
- Preservation of divergences: if M diverges, then $[\![M]\!]$ diverges.

where $[\![\cdot]\!]$ is the interpretation function, $\to^*$ means the reflexive and transitive closure of reduction in the target calculus, and $\simeq$ is an adequate equivalence relation. Milner checks precisely the two criteria above for his interpretation of the lazy λ-calculus [23] (where reductions are not allowed to take place under the λ-abstraction, nor in the right-hand side

[2] For example, since the factorial function *fac* is defined recursively, *fac n* = *if n* = 0 *then* 1 *else fac* $(n-1)$ 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.

of application)[3]. Sangiorgi and Walker [27] 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

$$\overline{v\vec{x}}(\llbracket \lambda y.R \rrbracket_{\mathbb{I}}^{\mathbb{M}}\, u \mid \llbracket \overrightarrow{x := N} \rrbracket_{\mathbb{I}}^{\mathbb{M}}) \;=\; \overline{v\vec{x}}(u(y).u(v).\llbracket R \rrbracket_{\mathbb{I}}^{\mathbb{M}}\, v \mid \overrightarrow{!\,x(w).\llbracket N \rrbracket_{\mathbb{I}}^{\mathbb{M}}\, w})$$

no communication is possible inside $\llbracket R \rrbracket_{\mathbb{I}}^{\mathbb{M}}\, v$ or $\llbracket N \rrbracket_{\mathbb{I}}^{\mathbb{M}}\, w$, 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 [7]), but rather large-step reduction, that reduces to lazy normal form directly; therefore, not all individual reduction steps are modelled. In fact, as argued in [7], it is not possible to show the first criterion for normal single step β-reduction for Milner's encoding, not even when restricted to lazy reduction. Departing from Milner's original result, most research in the direction of encodings into the π-calculus – *call-by-name*, *call-by-value* and *call-by-need* [27] – consider a λ-calculus where β-reductions are limited to lazy reduction; therefore, these encodings cannot model each single β-reduction step. The focus of Milner and later results has been the correspondence/relation between *observable behaviour* in the lazy λ-calculus and 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 $M \approx_c^{\lambda} N \iff \llbracket M \rrbracket \approx_c^{\pi} \llbracket N \rrbracket$, 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 [5, 7] 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 [7]; rather, we showed to faithfully represent *explicit spine reduction* (Def. 2.4) 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[4] 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 model β-equality also, which then gives that our interpretation gives, in fact, a semantics. We achieve our results by generalising the logical interpretation we presented in [7]. This interpretation is based on the encoding of implicative natural deduction into the sequent calculus, and induces a notion of implicative type assignment for the π-calculus, which we will use here as well.

Although our main objective is to study semantics for the λ-calculus, we *can* show a termination result, but only when restricting reduction in the π-calculus, where we do not allow reduction inside a *mute* process, *i.e.* a process with no free output name. We call this notion of reduction *lazy* as well; this solution to the termination problem corresponds to the restriction to lazy reduction for the λ-calculus when modelling the computable functions.

To be precise, we define an interpretation of λ-terms $\llbracket \cdot \rrbracket_{\mathbb{I}}^{\mathbb{F}} \cdot$, for which we will show[5]:

(*Operational Soundness*): $M \to_{\mathbf{x}} N \Rightarrow \llbracket M \rrbracket_{\mathbb{I}}^{\mathbb{F}} a \to_{\pi}^{*} \sim_{\mathrm{C}} \llbracket N \rrbracket_{\mathbb{I}}^{\mathbb{F}} a$;

[3] Milner also considers an interpretation that respects call-by-value reduction.

[4] 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.

[5] 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.

(*Adequacy*): $M =_\beta N \Rightarrow \llbracket M \rrbracket^{\mathrm{F}}_{\mathbb{I}} a \sim_{\mathrm{C}} \llbracket N \rrbracket^{\mathrm{F}}_{\mathbb{I}} a$;
(*Operational Completeness*): $\llbracket M \rrbracket^{\mathrm{F}}_{\mathbb{I}} a \rightarrow_\pi P \Rightarrow \exists N\, [P \rightarrow^*_\pi \sim_{\mathrm{C}} \llbracket N \rrbracket^{\mathrm{F}}_{\mathbb{I}} a \;\&\; M \rightarrow^*_{\mathbf{x}} N]$;
(*Type preservation*): $\Gamma \vdash_\lambda M : A \Rightarrow \llbracket M \rrbracket^{\mathrm{F}}_{\mathbb{I}} a : \Gamma \vdash^{io}_\pi a{:}A$;
(*Termination*): $M{\downarrow} \Rightarrow \llbracket M \rrbracket^{\mathrm{F}}_{\mathbb{I}} a {\downarrow_{\mathrm{L}\pi}}$.

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 [5, 7], and is different from other systems studied in the literature. 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 that we consider here.

Definition 1.1
- *Channel names* and *data* are defined by:

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

  Notice that pairing is *not* recursive.
- *Processes* are defined by the grammar (where x,y,z are *variables*):

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

- 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 $\textsf{let}\,\langle x,y\rangle = p\;\textsf{in}\;P$. We call n free in P if it occurs in P and is not bound; we write $fn(P)$ for the set of free names in P, and write $fn(P,Q)$ for $fn(P) \cup fn(Q)$.
- We call a process *mute* if it has no free output name.

Whether or not a process is mute is decidable.

We abbreviate $a(x).\textsf{let}\,\langle y,z\rangle = x\;\textsf{in}\;P$ by $a(y,z).P$, and $(\nu m)\,(\nu n)\,P$ by $(\nu mn)\,P$, and write $\overline{a}\,p$ for $\overline{a}\,p.\mathbf{0}$. Notice that all channels are monadic.

Definition 1.2 (Congruence) The structural congruence relation '$\equiv$' is the smallest equivalence relation closed under contexts defined by the following rules:

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

We will consider processes modulo congruence: this implies that we will not deal explicitly with the process $\textsf{let}\,\langle x,y\rangle = \langle a,b\rangle\;\textsf{in}\;P$, 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. We explicitly convert '*an output sent on a is to be received as input on b*' via '$a(w).\overline{b}\,w$' (called a *forwarder* in [20]), which, following [27], is abbreviated into $a \twoheadrightarrow b$.

Definition 1.3 (Reduction) *i*) The *reduction relation* over processes of the π-calculus is

defined by:

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

ii) We write $\rightarrow$ for the transitive closure of $\rightarrow_\pi$, and $\rightarrow_\pi^*$ for the reflexive, transitive closure of $\rightarrow_\pi$.

iii) We define $\rightarrow_{\mathrm{L}\pi}$ (*lazy* reduction) as $\rightarrow_\pi$, but do not allow reduction inside mute processes[6].

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

The following notions are standard, and of use:

Definition 1.4 *i*) We write $P{\downarrow}n$ (*P outputs on n*) if $P \equiv (\nu b_1 \ldots b_m)\,(\overline{n}\,p \mid Q)$ for some Q, where $n \notin \{b_1,\ldots,b_m\}$.

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

iii) 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 $\mathsf{C}[Q]{\Downarrow}n$.

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

The π-calculus is equipped with a rich type theory [27], from the basic type system for counting the arity of channels [25] to sophisticated linear types in [21], which studies a relation between Call-by-Value $\lambda\mu$ [24] and a linear π-calculus. The notion of type assignment we use here (first defined in [5]) 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 1.5 (Types and Contexts) *i*) Types are defined by: $A,B ::= \varphi \mid A{\rightarrow}B$ where φ is a basic type of which there are infinitely many.

ii) 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 $\Gamma, \{n{:}A\}$.

iii) 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 either $n{:}A \in \Gamma$, or Γ is not defined on n.

Definition 1.6 (Context assignment for π [5]) Functional type assignment for the π-calculus is defined by the following sequent system[7]:

---

[6] Lazy reduction can be seen as similar to garbage collection, since it ignores processes that produce no visible output.

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

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

$$(!):\ \frac{P : \Gamma \vdash^{io}_{\pi} \Delta}{!P : \Gamma \vdash^{io}_{\pi} \Delta} \qquad (out):\ \frac{P : \Gamma, b{:}A \vdash^{io}_{\pi} b{:}A, \Delta}{\overline{a}\, b : \Gamma, b{:}A \vdash^{io}_{\pi} a{:}A, b{:}A, \Delta}\ (a \neq b)$$

$$(\nu):\ \frac{P : \Gamma, a{:}A \vdash^{io}_{\pi} a{:}A, \Delta}{(\nu a)\, P : \Gamma \vdash^{io}_{\pi} \Delta}\ (a \notin \Gamma, \Delta) \qquad (in):\ \frac{P : \Gamma, x{:}A \vdash^{io}_{\pi} x{:}A, \Delta}{a(x).P : \Gamma, a{:}A \vdash^{io}_{\pi} \Delta}$$

$$(|):\ \frac{P_1 : \Gamma \vdash^{io}_{\pi} \Delta \quad \cdots \quad P_n : \Gamma \vdash^{io}_{\pi} \Delta}{P_1 \mid \cdots \mid P_n : \Gamma \vdash^{io}_{\pi} \Delta} \qquad (let):\ \frac{P : \Gamma, y{:}B \vdash^{io}_{\pi} x{:}A, \Delta}{let\ \langle x,y\rangle = z\ in\ P : \Gamma, z{:}A{\to}B \vdash^{io}_{\pi} \Delta}\ \begin{array}{l}(y,z \notin \Delta;\\ x,z \notin \Gamma)\end{array}$$

As usual, we write $P : \Gamma \vdash^{io}_{\pi} \Delta$ 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; handling of arrow types is restricted to the rules (let) and $(\langle\rangle\text{-}out)$.

The above system is not trivial, since the process

$$(\nu cb)\,(x(w).\overline{c}\,w \mid c(v,d).(!\,b{\to}v \mid d{\to}a) \mid !^{\mathbb{F}}\, x^{\mathbb{F}}_{\mathbb{J}}\, b)$$

(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 1.7 We can derive

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

so the following rule is derivable:

$$(\langle\rangle\text{-}in):\ \frac{P : \Gamma, y{:}B \vdash^{io}_{\pi} x{:}A, \Delta}{a(x,y).P : \Gamma, a{:}A{\to}B \vdash^{io}_{\pi} \Delta}\ (y,a) \notin \Delta, x \notin \Gamma$$

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

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

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

2 The λ-calculus and $\lambda\mathbf{x}$

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 $\lambda\mathbf{x}$ [10], a version of the λ-calculus with *explicit substitution*, and show our results for $\lambda\mathbf{x}$; since $\lambda\mathbf{x}$ implements β-reduction, we also show our results for normal β-reduction.

Definition 2.1 (Lambda terms and β-reduction [8])

i) The set Λ of *λ-terms* is defined by the grammar: $M, N \ ::= \ x \mid \lambda x.M \mid MN$.

ii) 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; $=_\beta$ is the smallest equivalence relation that contains $\to_\beta$.

iii) *Lazy* reduction $\to_{\mathrm{L}}$ [3] for the λ-calculus is defined by limiting $\to_\beta$ to:

$$(\lambda x.M)N \ \to\ M[N/x] \qquad M \to N \ \Rightarrow\ ML \to NL$$

iv) *Spine* reduction $\to_{\mathrm{s}}$ [7] 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-calculus 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 2.2 (C.F. [10]) *i*) The syntax of the *explicit lambda calculus* $\lambda\mathbf{x}$ is defined by:

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

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

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

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

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.

Proposition 2.3 ($\lambda\mathbf{x}$ IMPLEMENTS β-REDUCTION) • $M \to_\beta N \Rightarrow M \to_{\mathbf{x}}^* N$.
• $M \in \lambda$ & $M \to_{\mathbf{x}} N \Rightarrow \exists L \in \Lambda\ [N \to_{\mathbf{x}}^* L$ & $M \to_\beta^* L]$.

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).

In [7], we defined two restrictions of reduction in $\lambda\mathbf{x}$ which are useful for the results of this paper.

Definition 2.4 ($\lambda\mathbf{x}$L AND $\lambda\mathbf{x}$S [7]) We define two sub-reduction systems of $\lambda\mathbf{x}$:

i) *Explicit lazy reduction* $\to_{\mathbf{x}\mathrm{L}}$ is defined as follows.

$$\begin{array}{rcll} (\lambda x.M)N & \to & M\langle x := N\rangle & \\ M\langle x := N\rangle & \to & M & (x \notin fv(M)) \\ ((xM_1\cdots M_n)\,\overrightarrow{\langle y := L\rangle})\langle x := N\rangle & \to & ((NM_1\cdots M_n)\,\overrightarrow{\langle y := L\rangle})\langle x := N\rangle & \end{array}$$

$$M \to N \ \Rightarrow\ \begin{cases} ML & \to\ NL \\ M\langle x := L\rangle & \to\ N\langle x := L\rangle \end{cases}$$

ii) *Explicit spine reduction* $\to_{\mathbf{xs}}$ is defined as $\to_{\mathbf{xl}}$, but by adding

$$\begin{array}{rcl}(\lambda y.M)\langle x:=N\rangle & \to & \lambda y.(M\langle x:=N\rangle)\\ M\to_\pi N & \Rightarrow & \lambda x.M\to_\pi \lambda x.N\end{array}$$

The difference between these two notions is that explicit spine reduction allows a substitution to propagate under an abstraction, and that explicit spine reduction can take place under an abstraction as well.

In both these notions, substitution is lazy: notice that we are reducing only at the head of a term, and only when the variable mentioned in the pending substitution appears in the head will that substitution be executed; this appears to be the implicit approach of [23] (see Lemma 4.5, case 3).

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 2.5 Using the notion of types in Def. 1.5, type assignment for $\lambda\mathbf{x}$ is defined by:

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

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

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

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

Definition 3.1 (Full logical interpretation)

$$\begin{array}{rcl}\mathbb{[}x\mathbb{]}^{\mathrm{F}}_{\mathbb{]}}a & \triangleq & x(w).\bar{a}\,w\\ \mathbb{[}\lambda x.M\mathbb{]}^{\mathrm{F}}_{\mathbb{]}}a & \triangleq & (\nu xb)\,(\mathbb{[}M\mathbb{]}^{\mathrm{F}}_{\mathbb{]}}b\mid \bar{a}\langle x,b\rangle)\\ \mathbb{[}MN\mathbb{]}^{\mathrm{F}}_{\mathbb{]}}a & \triangleq & (\nu cb)\,(\mathbb{[}M\mathbb{]}^{\mathrm{F}}_{\mathbb{]}}c\mid c(v,d).(!\,b\twoheadrightarrow v\mid d\twoheadrightarrow a)\mid !\mathbb{[}N\mathbb{]}^{\mathrm{F}}_{\mathbb{]}}b)\\ \mathbb{[}M\langle x:=N\rangle\mathbb{]}^{\mathrm{F}}_{\mathbb{]}}a & \triangleq & (\nu x)\,(\mathbb{[}M\mathbb{]}^{\mathrm{F}}_{\mathbb{]}}a\mid !\mathbb{[}N\mathbb{]}^{\mathrm{F}}_{\mathbb{]}}x)\end{array}$$

Notice that we use of $a\twoheadrightarrow b$; we write $a\twoheadrightarrow b$ for auxiliary forwarders generated by the interpretation, and $x(w).\bar{a}\,w$ for the interpretation of x under the channel name a.

The replicated term in the interpretation of the substitution is not guarded, so can run on its own; this in contrast to the case for Milner's encoding (Def. 4.1), 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;

• 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* $c(v,d).(!\,b\twoheadrightarrow v\mid d\twoheadrightarrow a)$; the received input v name is used to redirect the output for N arriving over b (since $\mathbb{[}N\mathbb{]}^{\mathrm{F}}_{\mathbb{]}}b$ gets replicated, so does $b\twoheadrightarrow v$) and 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 $\llbracket MN \rrbracket^{\mathrm{F}}_{\mathrm{J}} a$ in terms of $\llbracket M \rrbracket^{\mathrm{F}}_{\mathrm{J}} b$ and $\llbracket N \rrbracket^{\mathrm{F}}_{\mathrm{J}} c$, 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

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

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 ($\lambda\mathbf{x}$ [10]) 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 $(\nu ybzb_1)\,(y(w).\overline{b_1}\,w \mid \overline{b}\langle z,b_1\rangle \mid \overline{a}\langle y,b\rangle)$ and $(\nu xb)\,(x(w).\overline{b}\,w \mid \overline{a}\langle x,b\rangle)$, respectively, processes that differ under $\sim_{\mathrm{C}}$.

Even though our interpretation $\llbracket \cdot \rrbracket^{\mathrm{F}}_{\mathrm{J}} \cdot$ is fundamentally different form the one we presented in [7] (see Def. 4.3), we can still show that typeability is preserved:

Theorem 3.2 ($\llbracket \cdot \rrbracket^{\mathrm{F}}_{\mathrm{J}} \cdot$ PRESERVES CURRY TYPES) *If* $\Gamma \vdash_\lambda M : A$, *then* $\llbracket M \rrbracket^{\mathrm{F}}_{\mathrm{J}} a : \Gamma \vdash^{io}_{\pi} a : A$.

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 $\llbracket x \rrbracket^{\mathrm{F}}_{\mathrm{J}} a = x(w).\overline{a}\,w$, and that

$$\dfrac{\dfrac{}{\overline{a}\,w : \Gamma', w{:}A \vdash^{io}_{\pi} a{:}A, w{:}A}\,(out)}{x(w).\overline{a}\,w : \Gamma', x{:}A \vdash^{io}_{\pi} a{:}A}\,(in)$$

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

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

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

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{v}\,w : \Gamma, w{:}B \vdash^{io}_{\pi} v{:}B, w{:}B, \Delta}\,(out)}{b \twoheadrightarrow v : b{:}B \vdash^{io}_{\pi} v{:}B}\,(in)}{!\,b \twoheadrightarrow v : b{:}B \vdash^{io}_{\pi} v{:}B}\,(!) \qquad \dfrac{\dfrac{}{\overline{a}\,w : \Gamma, w{:}A \vdash^{io}_{\pi} a{:}A, w{:}A, \Delta}\,(out)}{d \twoheadrightarrow a : d{:}A \vdash^{io}_{\pi} a{:}A}\,(in)}{!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a : \Gamma, b{:}B, d{:}A \vdash^{io}_{\pi} v{:}B, a{:}A}\,(|) \qquad \boxed{\mathcal{D}_2}}{c\,(v,d).\,(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a) : \Gamma, b{:}B, c{:}B \to A \vdash^{io}_{\pi} a{:}A}\,(\langle\rangle\text{-}in) \qquad \dfrac{\mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,b : \Gamma \vdash^{io}_{\pi} b{:}B}{!\,\mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,b : \Gamma \vdash^{io}_{\pi} b{:}B}\,(!)}{\dfrac{\mathbb{F}\,P^{\mathrm{F}}_{\Downarrow}\,c : \Gamma \vdash^{io}_{\pi} c{:}B \to A \qquad \mathbb{F}\,P^{\mathrm{F}}_{\Downarrow}\,c \mid c\,(v,d).\,(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\,\mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,b : \Gamma, b{:}B, c{:}B \to A \vdash^{io}_{\pi} c{:}B \to A, a{:}A}{(\nu b)\,(\mathbb{F}\,P^{\mathrm{F}}_{\Downarrow}\,c \mid c\,(v,d).\,(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\,\mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,b) : \Gamma, c{:}B \to A \vdash^{io}_{\pi} c{:}B \to A, a{:}A}\,(|)}{(\nu cb)\,(\mathbb{F}\,P^{\mathrm{F}}_{\Downarrow}\,c \mid c\,(v,d).\,(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\,\mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,b) : \Gamma \vdash^{io}_{\pi} a{:}A}\,(\nu)}\,(\nu)$$

$\boxed{\mathcal{D}_1}$ $\mathbb{F}\,P^{\mathrm{F}}_{\Downarrow}\,c : \Gamma \vdash^{io}_{\pi} c{:}B \to A$

(*cut*): Then $M = P\,\langle x := Q\rangle$, and there exists B such that $\Gamma, x{:}B \vdash_{\lambda} P : A$ and $\Gamma \vdash_{\lambda} Q : B$, and $\mathbb{F}\,P\,\langle x := Q\rangle^{\mathrm{F}}_{\Downarrow}\,a = (\nu x)\,(\mathbb{F}\,P^{\mathrm{F}}_{\Downarrow}\,a \mid !\,\mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,x)$. By induction, there exist $\mathcal{D}_1 :: \mathbb{F}\,P^{\mathrm{F}}_{\Downarrow}\,c : \Gamma, x{:}B \vdash^{io}_{\pi} a{:}A$ and $\mathcal{D}_2 :: \mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,x : \Gamma \vdash^{io}_{\pi} x{:}B$, and we can construct:

$$\dfrac{\dfrac{\boxed{\mathcal{D}_1} \qquad \dfrac{\boxed{\mathcal{D}_2}}{\mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,x : \Gamma \vdash^{io}_{\pi} x{:}B}}{\mathbb{F}\,P^{\mathrm{F}}_{\Downarrow}\,a : \Gamma, x{:}B \vdash^{io}_{\pi} a{:}A \qquad !\,\mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,x : \Gamma \vdash^{io}_{\pi} x{:}B}\,(!)}{\mathbb{F}\,P^{\mathrm{F}}_{\Downarrow}\,a \mid !\,\mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,x : \Gamma, x{:}B \vdash^{io}_{\pi} x{:}B, a{:}A}\,(|)}{(\nu x)\,(\mathbb{F}\,P^{\mathrm{F}}_{\Downarrow}\,a \mid !\,\mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,x) : \Gamma \vdash^{io}_{\pi} a{:}A}\,(\nu)$$

$\boxed{\mathcal{D}_1}$ $\mathbb{F}\,P^{\mathrm{F}}_{\Downarrow}\,a : \Gamma, x{:}B \vdash^{io}_{\pi} a{:}A$ $\boxed{\mathcal{D}_2}$ $\mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,x : \Gamma \vdash^{io}_{\pi} x{:}B$

□

Example 3.3 The interpretation of a redex reduces as:

$$\begin{array}{ll}
\mathbb{F}(\lambda x.P)Q^{\mathrm{F}}_{\Downarrow}\,a & \stackrel{\Delta}{=} \\
(\nu cb)\,((\nu x b_1)\,(\mathbb{F}\,P^{\mathrm{F}}_{\Downarrow}\,b_1 \mid \overline{c}\langle x, b_1\rangle) \mid c\,(v,d).\,(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\,\mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,b) & \to_{\pi} (c) \\
(\nu b x b_1)\,(\mathbb{F}\,P^{\mathrm{F}}_{\Downarrow}\,b_1 \mid !\,b \twoheadrightarrow x \mid b_1 \twoheadrightarrow a \mid !\,\mathbb{F}\,Q^{\mathrm{F}}_{\Downarrow}\,b) &
\end{array}$$

So reduction in π implements β-reduction by at least performing this step; this implies that we model each β-reduction step by at least one π-reduction.

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

$$\begin{array}{ll}
\mathbb{F}(\lambda x.(\lambda z.(\lambda y.M)x))N^{\mathrm{F}}_{\Downarrow}\,a & \stackrel{\Delta}{=}, \equiv \\
\quad (\nu cb)\,((\nu x b_1)\,(\mathbb{F}\lambda z.(\lambda y.M)x^{\mathrm{F}}_{\Downarrow}\,b_1 \mid \overline{c}\langle x, b_1\rangle) \mid c\,(v,d).\,(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\,\mathbb{F}N^{\mathrm{F}}_{\Downarrow}\,b) & \to_{\pi} (c) \\
\quad (\nu b)\,((\nu x b_1)\,((\nu z b_2)\,((\nu c_1 b_3)\,((\nu y b_4)\,(\mathbb{F}\,M^{\mathrm{F}}_{\Downarrow}\,b_4 \mid \overline{c_1}\langle y, b_4\rangle) \mid & \\
\qquad c_1\,(v,d).\,(!\,b_3 \twoheadrightarrow v \mid d \twoheadrightarrow b_2) \mid !\,\mathbb{F}\,x^{\mathrm{F}}_{\Downarrow}\,b_3) \mid \overline{b_1}\langle z, b_2\rangle) \mid !\,b \twoheadrightarrow x \mid b_1 \twoheadrightarrow a) \mid !\,\mathbb{F}N^{\mathrm{F}}_{\Downarrow}\,b) & \to_{\pi} (c_1) \\
\quad (\nu b)\,((\nu x b_1)\,((\nu z b_2)\,((\nu b_3)\,((\nu y b_4)\,(\mathbb{F}\,M^{\mathrm{F}}_{\Downarrow}\,b_4 \mid !\,b_3 \twoheadrightarrow y \mid b_4 \twoheadrightarrow b_2) \mid !\,\mathbb{F}\,x^{\mathrm{F}}_{\Downarrow}\,b_3) \mid & \\
\qquad \overline{b_1}\langle z, b_2\rangle) \mid !\,b \twoheadrightarrow x \mid b_1 \twoheadrightarrow a) \mid !\,\mathbb{F}N^{\mathrm{F}}_{\Downarrow}\,b) & \to_{\pi} (b_1) \\
\quad (\nu z b_2)\,((\nu y b_4)\,(\mathbb{F}\,M^{\mathrm{F}}_{\Downarrow}\,b_4 \mid b_4 \twoheadrightarrow b_2 \mid & \\
\qquad (\nu b x b_3)\,(!\,b_3 \twoheadrightarrow y \mid !\,\mathbb{F}\,x^{\mathrm{F}}_{\Downarrow}\,b_3 \mid !\,b \twoheadrightarrow x \mid !\,\mathbb{F}N^{\mathrm{F}}_{\Downarrow}\,b)) \mid \overline{a}\langle z, b_2\rangle) & \sim_{\mathrm{C}} (b, x, b_3) \\
\quad (\nu z b_2)\,((\nu y b_4)\,(\mathbb{F}\,M^{\mathrm{F}}_{\Downarrow}\,b_4 \mid b_4 \twoheadrightarrow b_2 \mid !\,\mathbb{F}N^{\mathrm{F}}_{\Downarrow}\,y) \mid \overline{a}\langle z, b_2\rangle) & \sim_{\mathrm{C}} (b_4) \\
\quad (\nu z b_2)\,((\nu y)\,(\mathbb{F}\,M^{\mathrm{F}}_{\Downarrow}\,b_2 \mid !\,\mathbb{F}N^{\mathrm{F}}_{\Downarrow}\,y) \mid \overline{a}\langle z, b_2\rangle) & \stackrel{\Delta}{=} \mathbb{F}\lambda z.(M\,\langle y := N\rangle)^{\mathrm{F}}_{\Downarrow}\,a
\end{array}$$

Lemma 3.5 *The following are admissible.*

i) $!\,P \equiv !\,P \mid !\,P$.

ii) $(\nu a)\,((\nu b)\,(Q \mid R) \mid !\,P) \;\equiv\; (\nu b)\,((\nu a)\,(Q \mid !\,P) \mid (\nu a)\,(R \mid !\,P))$ *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).*

iii) $(\nu x)\,(!\,\mathbb{F}N^{\mathrm{F}}_{\Downarrow}\,b \mid !\,\mathbb{F}P^{\mathrm{F}}_{\Downarrow}\,x) \sim_{\mathrm{C}} !\,((\nu x)\,\mathbb{F}N^{\mathrm{F}}_{\Downarrow}\,b \mid !\,\mathbb{F}P^{\mathrm{F}}_{\Downarrow}\,x)$.

Proof: Easy. □

We will now show that our interpretation fully respects the explicit reduction $\to_{\mathbf{x}}$, 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.

Proposition 3.6 $(\nu xb)\,(c(v,d).(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow e)) \sim_{\mathrm{C}} (\nu a)\,(a \twoheadrightarrow e \mid (\nu xb)\,(c(v,d).(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a)))$.

We use this result to show the following:

Lemma 3.7 (Renaming lemma) $(\nu a)\,(a \twoheadrightarrow e \mid \llbracket M \rrbracket^{\mathrm{F}}_{\parallel} a) \;\sim_{\mathrm{C}}\; \llbracket M \rrbracket^{\mathrm{F}}_{\parallel} e$.

Proof: *i)* $(\nu a)\,(a \twoheadrightarrow e \mid \llbracket x \rrbracket^{\mathrm{F}}_{\parallel} a) \;\triangleq\; (\nu a)\,(a \twoheadrightarrow e \mid x(w).\bar{a}\,w) \;\sim_{\mathrm{C}}\; x(w).\bar{e}\,w \;\triangleq\; \llbracket x \rrbracket^{\mathrm{F}}_{\parallel} e$

ii) $(\nu a)\,(a \twoheadrightarrow e \mid \llbracket \lambda x.M \rrbracket^{\mathrm{F}}_{\parallel} a) \;\triangleq\; (\nu a)\,(a \twoheadrightarrow e \mid (\nu xb)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} b \mid \bar{a}\langle x,b\rangle)) \;\equiv$
$(\nu axb)\,(a \twoheadrightarrow e \mid \llbracket M \rrbracket^{\mathrm{F}}_{\parallel} b \mid \bar{a}\langle x,b\rangle) \;\sim_{\mathrm{C}}\; (\nu xb)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} b \mid \bar{e}\langle x,b\rangle) \;\sim_{\mathrm{C}}\; \llbracket \lambda x.M \rrbracket^{\mathrm{F}}_{\parallel} e$

iii) $(\nu a)\,(a \twoheadrightarrow e \mid \llbracket MN \rrbracket^{\mathrm{F}}_{\parallel} a) \;\triangleq$
$(\nu a)\,(a \twoheadrightarrow e \mid (\nu cb)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} c \mid c(v,d).(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\llbracket N \rrbracket^{\mathrm{F}}_{\parallel} b)) \;\sim_{\mathrm{C}}\; (3.6)$
$(\nu cb)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} c \mid c(v,d).(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow e) \mid !\llbracket N \rrbracket^{\mathrm{F}}_{\parallel} b) \;\triangleq\; \llbracket MN \rrbracket^{\mathrm{F}}_{\parallel} e$

iv) $(\nu a)\,(a \twoheadrightarrow e \mid \llbracket M\langle x := N\rangle \rrbracket^{\mathrm{F}}_{\parallel} a) \;\triangleq\; (\nu a)\,(a \twoheadrightarrow e \mid (\nu x)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} a \mid !\llbracket N \rrbracket^{\mathrm{F}}_{\parallel} x)) \;\equiv$
$(\nu x)\,((\nu a)\,(a \twoheadrightarrow e \mid \llbracket M \rrbracket^{\mathrm{F}}_{\parallel} a) \mid !\llbracket N \rrbracket^{\mathrm{F}}_{\parallel} x) \;\sim_{\mathrm{C}}\; (IH)$
$(\nu x)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} e \mid !\llbracket N \rrbracket^{\mathrm{F}}_{\parallel} x) \;\triangleq\; \llbracket M\langle x := N\rangle \rrbracket^{\mathrm{F}}_{\parallel} e$ □

Using this lemma, we can show that:

$$(\nu bxb_1)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} b_1 \mid !\,b \twoheadrightarrow x \mid b_1 \twoheadrightarrow a \mid !\llbracket N \rrbracket^{\mathrm{F}}_{\parallel} b) \;\sim_{\mathrm{C}}\; (\nu x)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} a \mid !\llbracket N \rrbracket^{\mathrm{F}}_{\parallel} x)$$

which, in part, justifies the last case of Def. 3.1.

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

Lemma 3.8 $\llbracket P\langle y := Q\rangle\langle x := R\rangle \rrbracket^{\mathrm{F}}_{\parallel} a \;\sim_{\mathrm{C}}\; \llbracket P\langle x := R\rangle\langle y := Q\langle x := R\rangle\rangle \rrbracket^{\mathrm{F}}_{\parallel} a$.

Proof: $\llbracket P\langle y := Q\rangle\langle x := R\rangle \rrbracket^{\mathrm{F}}_{\parallel} a \;\triangleq\; (\nu x)\,((\nu y)\,(\llbracket P \rrbracket^{\mathrm{F}}_{\parallel} a \mid !\llbracket Q \rrbracket^{\mathrm{F}}_{\parallel} y) \mid !\llbracket R \rrbracket^{\mathrm{F}}_{\parallel} x) \;\equiv$
$(\nu xy)\,(\llbracket P \rrbracket^{\mathrm{F}}_{\parallel} a \mid !\llbracket Q \rrbracket^{\mathrm{F}}_{\parallel} y \mid !\llbracket R \rrbracket^{\mathrm{F}}_{\parallel} x) \;\sim_{\mathrm{C}}\; (\nu xy)\,(\llbracket P \rrbracket^{\mathrm{F}}_{\parallel} a \mid !\llbracket R \rrbracket^{\mathrm{F}}_{\parallel} x \mid !\llbracket Q \rrbracket^{\mathrm{F}}_{\parallel} y \mid !\llbracket R \rrbracket^{\mathrm{F}}_{\parallel} x) \;\sim_{\mathrm{C}}$
$(\nu y)\,((\nu x)\,(\llbracket P \rrbracket^{\mathrm{F}}_{\parallel} a \mid !\llbracket R \rrbracket^{\mathrm{F}}_{\parallel} x) \mid (\nu x)\,(!\llbracket Q \rrbracket^{\mathrm{F}}_{\parallel} y \mid !\llbracket R \rrbracket^{\mathrm{F}}_{\parallel} x)) \;\sim_{\mathrm{C}}$
$(\nu y)\,((\nu x)\,(\llbracket P \rrbracket^{\mathrm{F}}_{\parallel} a \mid !\llbracket R \rrbracket^{\mathrm{F}}_{\parallel} x) \mid !\llbracket Q\langle x := R\rangle \rrbracket^{\mathrm{F}}_{\parallel} y) \;\triangleq\; (\nu y)\,(\llbracket P\langle x := R\rangle \rrbracket^{\mathrm{F}}_{\parallel} a \mid !\llbracket (Q\langle x := R\rangle) \rrbracket^{\mathrm{F}}_{\parallel} y) \;\triangleq$
$\llbracket P\langle x := R\rangle\langle y := Q\langle x := R\rangle\rangle \rrbracket^{\mathrm{F}}_{\parallel} a$ □

As in [23, 27], we can now show a reduction preservation result for full explicit reduction for $\lambda\mathbf{x}$, by showing that $\llbracket \cdot \rrbracket^{\mathrm{F}}_{\parallel} \cdot$ preserves $\to_{\mathbf{x}}$ up to $\sim_{\mathrm{C}}$. As for Thm. 4.4, we do not require the terms to be closed:

Theorem 3.9 (Preservation of reduction) $M \to_{\mathbf{x}} N \Rightarrow \llbracket M \rrbracket^{\mathrm{F}}_{\parallel} a \sim_{\mathrm{C}} \llbracket N \rrbracket^{\mathrm{F}}_{\parallel} a$.

Proof: By induction on explicit reduction.

$((\lambda x.M)P \to_{\pi} M\langle x := P\rangle)$: $\llbracket (\lambda x.M)P \rrbracket^{\mathrm{F}}_{\parallel} a \;\triangleq$
$(\nu cb)\,((\nu xb_1)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} b_1 \mid \bar{c}\langle x,b_1\rangle) \mid c(v,d).(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\llbracket P \rrbracket^{\mathrm{F}}_{\parallel} b) \;\sim_{\mathrm{C}}\; (c)$
$(\nu cb)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} b_1 \mid !\,b \twoheadrightarrow x \mid b_1 \twoheadrightarrow a \mid !\llbracket P \rrbracket^{\mathrm{F}}_{\parallel} b) \;\sim_{\mathrm{C}}\; (\nu x)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} a \mid !\llbracket P \rrbracket^{\mathrm{F}}_{\parallel} x)$

$((MN)\langle x := P\rangle \to_{\pi} M\langle x := P\rangle N\langle x := P\rangle)$: $\llbracket MN\langle x := P\rangle \rrbracket^{\mathrm{F}}_{\parallel} a \;\triangleq$
$(\nu x)\,(\llbracket MN \rrbracket^{\mathrm{F}}_{\parallel} a \mid !\llbracket P \rrbracket^{\mathrm{F}}_{\parallel} x) \;\triangleq$
$(\nu x)\,((\nu cb)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} c \mid c(v,d).(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\llbracket N \rrbracket^{\mathrm{F}}_{\parallel} b) \mid !\llbracket P \rrbracket^{\mathrm{F}}_{\parallel} x) \;\sim_{\mathrm{C}}\; (3.5(ii))$
$(\nu cb)\,((\nu x)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} c \mid !\llbracket P \rrbracket^{\mathrm{F}}_{\parallel} x) \mid c(v,d).(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid (\nu x)\,(!\llbracket N \rrbracket^{\mathrm{F}}_{\parallel} b \mid !\llbracket P \rrbracket^{\mathrm{F}}_{\parallel} x)) \;\sim_{\mathrm{C}}\; (3.5(iii))$
$(\nu cb)\,((\nu x)\,(\llbracket M \rrbracket^{\mathrm{F}}_{\parallel} c \mid !\llbracket P \rrbracket^{\mathrm{F}}_{\parallel} x) \mid c(v,d).(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\llbracket N\langle x := P\rangle \rrbracket^{\mathrm{F}}_{\parallel} b) \;\triangleq$
$(\nu cb)\,(\llbracket M\langle x := P\rangle \rrbracket^{\mathrm{F}}_{\parallel} c \mid c(v,d).(!\,b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\llbracket N\langle x := P\rangle \rrbracket^{\mathrm{F}}_{\parallel} b) \;\triangleq$
$\llbracket M\langle x := P\rangle N\langle x := P\rangle \rrbracket^{\mathrm{F}}_{\parallel} a$

$((\lambda y.M)\langle x:=P\rangle \to_\pi \lambda y.(M\langle x:=P\rangle))$: $\quad \llbracket(\lambda y.M)\langle x:=P\rangle\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \quad \stackrel{\Delta}{=}$
$(\nu x)((\nu y b)(\llbracket M\rrbracket^{\mathrm{F}}_{\mathbb{I}} b \mid \overline{a}\langle y,b\rangle) \mid !\llbracket P\rrbracket^{\mathrm{F}}_{\mathbb{I}} x) \equiv (\nu y b)((\nu x)(\llbracket M\rrbracket^{\mathrm{F}}_{\mathbb{I}} b \mid !\llbracket P\rrbracket^{\mathrm{F}}_{\mathbb{I}} x) \mid \overline{a}\langle y,b\rangle) \stackrel{\Delta}{=}$
$\llbracket\lambda y.M\langle x:=P\rangle\rrbracket^{\mathrm{F}}_{\mathbb{I}} a$

$(x\langle x:=P\rangle \to_\pi P)$: $\quad \llbracket x\langle x:=P\rangle\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \stackrel{\Delta}{=} (\nu x)(\llbracket x\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \mid !\llbracket P\rrbracket^{\mathrm{F}}_{\mathbb{I}} x) \equiv$
$(\nu x)(x(w).\overline{a}\,w \mid \llbracket P\rrbracket^{\mathrm{F}}_{\mathbb{I}} x \mid !\llbracket P\rrbracket^{\mathrm{F}}_{\mathbb{I}} x) \sim_{\mathrm{C}} (3.7) \quad (\nu x)(\llbracket P\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \mid !\llbracket P\rrbracket^{\mathrm{F}}_{\mathbb{I}} x) \equiv$
$\llbracket P\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \mid (\nu x)(!\llbracket P\rrbracket^{\mathrm{F}}_{\mathbb{I}} x) \sim_{\mathrm{C}} \llbracket P\rrbracket^{\mathrm{F}}_{\mathbb{I}} a$

$(y\langle x:=P\rangle \to_\pi y)$: $\quad \llbracket y\langle x:=P\rangle\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \stackrel{\Delta}{=} (\nu x)(\llbracket y\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \mid !\llbracket P\rrbracket^{\mathrm{F}}_{\mathbb{I}} x) \equiv \llbracket y\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \mid (\nu x)(!\llbracket P\rrbracket^{\mathrm{F}}_{\mathbb{I}} x) \sim_{\mathrm{C}} \llbracket y\rrbracket^{\mathrm{F}}_{\mathbb{I}} a$

$(M \to_\pi N \Rightarrow ML \to_\pi NL)$: $\quad \llbracket ML\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \stackrel{\Delta}{=}$
$(\nu c b)(\llbracket M\rrbracket^{\mathrm{F}}_{\mathbb{I}} c \mid c(v,d).(!b\to v \mid d\to a) \mid !\llbracket L\rrbracket^{\mathrm{F}}_{\mathbb{I}} b) \sim_{\mathrm{C}} (IH)$
$(\nu c b)(\llbracket N\rrbracket^{\mathrm{F}}_{\mathbb{I}} c \mid c(v,d).(!b\to v \mid d\to a) \mid !\llbracket L\rrbracket^{\mathrm{F}}_{\mathbb{I}} b) \stackrel{\Delta}{=} \llbracket NL\rrbracket^{\mathrm{F}}_{\mathbb{I}} a$

$(M \to_\pi N \Rightarrow LM \to_\pi LN)$: $\quad \llbracket LM\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \stackrel{\Delta}{=}$
$(\nu c b)(\llbracket L\rrbracket^{\mathrm{F}}_{\mathbb{I}} c \mid c(v,d).(!b\to v \mid d\to a) \mid !\llbracket M\rrbracket^{\mathrm{F}}_{\mathbb{I}} b) \sim_{\mathrm{C}} (IH)$
$(\nu c b)(\llbracket L\rrbracket^{\mathrm{F}}_{\mathbb{I}} c \mid c(v,d).(!b\to v \mid d\to a) \mid !\llbracket N\rrbracket^{\mathrm{F}}_{\mathbb{I}} b) \stackrel{\Delta}{=} \llbracket LN\rrbracket^{\mathrm{F}}_{\mathbb{I}} a$

$(M \to_\pi N \Rightarrow \lambda x.M \to_\pi \lambda x.N)$: $\quad \llbracket\lambda x.M\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \stackrel{\Delta}{=} (\nu x b)(\llbracket M\rrbracket^{\mathrm{F}}_{\mathbb{I}} b \mid \overline{a}\langle x,b\rangle) \sim_{\mathrm{C}} (IH)$
$(\nu x b)(\llbracket N\rrbracket^{\mathrm{F}}_{\mathbb{I}} b \mid \overline{a}\langle x,b\rangle) \stackrel{\Delta}{=} \llbracket\lambda x.N\rrbracket^{\mathrm{F}}_{\mathbb{I}} a$

$(M \to_\pi N \Rightarrow M\langle x:=L\rangle \to_\pi N\langle x:=L\rangle)$: $\quad \llbracket M\langle x:=L\rangle\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \stackrel{\Delta}{=}$
$(\nu x)(\llbracket M\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \mid !\llbracket L\rrbracket^{\mathrm{F}}_{\mathbb{I}} x) \sim_{\mathrm{C}} (IH) \quad (\nu x)(\llbracket N\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \mid !\llbracket L\rrbracket^{\mathrm{F}}_{\mathbb{I}} x) \stackrel{\Delta}{=} \llbracket N\langle x:=L\rangle\rrbracket^{\mathrm{F}}_{\mathbb{I}} a$

$(M \to_\pi N \Rightarrow L\langle x:=M\rangle \to_\pi L\langle x:=N\rangle)$: $\quad \llbracket L\langle x:=M\rangle\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \stackrel{\Delta}{=} (\nu x)(\llbracket L\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \mid !\llbracket M\rrbracket^{\mathrm{F}}_{\mathbb{I}} x) \sim_{\mathrm{C}} (IH)$
$(\nu x)(\llbracket L\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \mid !\llbracket N\rrbracket^{\mathrm{F}}_{\mathbb{I}} x) \stackrel{\Delta}{=} \llbracket L\langle x:=N\rangle\rrbracket^{\mathrm{F}}_{\mathbb{I}} a$

□

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 3.10 (Operational Soundness for $\to_\beta$) *i)* $M \to_\beta^* N \Rightarrow \llbracket M\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \sim_{\mathrm{C}} \llbracket N\rrbracket^{\mathrm{F}}_{\mathbb{I}} a$.
ii) $M{\uparrow} \Rightarrow \llbracket M\rrbracket^{\mathrm{F}}_{\mathbb{I}} a{\uparrow}$.

The first is shown by induction using Thm. 3.9; the second follows from Ex. 3.3.

Since $\sim_{\mathrm{C}}$ is symmetric, Thm. 3.10 immediately gives that $\llbracket\cdot\rrbracket^{\mathrm{F}}_{\mathbb{I}}\cdot$ preserves $=_\beta$ up to $\sim_{\mathrm{C}}$, which states that our interpretation gives, in fact, a semantics for the λ-calculus.

Corollary 3.11 (Adequacy) *If* $M =_\beta N$, *then* $\llbracket M\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \sim_{\mathrm{C}} \llbracket N\rrbracket^{\mathrm{F}}_{\mathbb{I}} a$.

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

Theorem 3.12 (Operational Completeness for $\to_\beta$ and $\to_{\mathbf{x}}$) *i) Let* M *be a term in* $\lambda\mathbf{x}$. *If* $\llbracket M\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \to_\pi P$ *then there is* N *such that* $P \sim_{\mathrm{C}} \llbracket N\rrbracket^{\mathrm{F}}_{\mathbb{I}} a$, *and* $M \to_{\mathbf{x}}^* N$.
ii) Let $M \in \Lambda$, i.e. *a (pure)* λ*-term. If* $\llbracket M\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \to_\pi P$ *then there exists* $N \in \Lambda$ *such that* $P \sim_{\mathrm{C}} \llbracket N\rrbracket^{\mathrm{F}}_{\mathbb{I}} a$, *and* $M \to_\beta^* N$.

We cannot show "*if* $\llbracket M\rrbracket^{\mathrm{F}}_{\mathbb{I}} a \sim_{\mathrm{C}} \llbracket N\rrbracket^{\mathrm{F}}_{\mathbb{I}} a$, *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 $\llbracket\cdot\rrbracket^{\mathrm{F}}_{\mathbb{I}}\cdot$ also never exhibit an output (see the third example in Ex. 3.13). It seems that $=_{\mathrm{BT}}$ (which equates terms that have the same Böhm tree) is suitable; we will investigate this in future work.

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

Example 3.13 The interpretation of an unsolvable term has no observable output:

$$\begin{array}{ll}
\llbracket \Delta\Delta \rrbracket^{\mathrm{F}}_{\parallel} a \triangleq (\nu cb)\,((\nu xb_1)\,(\llbracket xx \rrbracket^{\mathrm{F}}_{\parallel} b_1 \mid \overline{c}\langle x,b_1\rangle) \mid c(v,d).(!\, b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\parallel} b) & \to_{\pi} (c) \\
(\nu b)\,((\nu xb_1)\,(\llbracket xx \rrbracket^{\mathrm{F}}_{\parallel} b_1 \mid !\, b \twoheadrightarrow x \mid b_1 \twoheadrightarrow a) \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\parallel} b) & \equiv, \triangleq \\
(\nu b)\,((\nu xb_1)\,((\nu c_1 b_2)\,(x(w).\overline{c_1}\, w \mid c_1(v,d).(!\, b_2 \twoheadrightarrow v \mid d \twoheadrightarrow b_1) \mid !\llbracket x \rrbracket^{\mathrm{F}}_{\parallel} b_2) \mid & \\
\quad !\, b \twoheadrightarrow x \mid b_1 \twoheadrightarrow a) \mid (\nu y b_3)\,(\llbracket yy \rrbracket^{\mathrm{F}}_{\parallel} b_3 \mid \overline{b}\langle y,b_3\rangle) \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\parallel} b) & \to_{\pi} (b,x) \\
(\nu b)\,((\nu xb_1)\,((\nu c_1 b_2)\,((\nu y b_3)\,(\llbracket yy \rrbracket^{\mathrm{F}}_{\parallel} b_3 \mid \overline{c_1}\langle y,b_3\rangle) \mid & \\
\quad c_1(v,d).(!\, b_2 \twoheadrightarrow v \mid d \twoheadrightarrow b_1) \mid !\llbracket x \rrbracket^{\mathrm{F}}_{\parallel} b_2) \mid !\, b \twoheadrightarrow x \mid b_1 \twoheadrightarrow a) \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\parallel} b) & \triangleq \\
(\nu b)\,((\nu xb_1)\,((\nu c_1 b_2)\,(\llbracket \Delta \rrbracket^{\mathrm{F}}_{\parallel} c_1 \mid & \\
\quad c_1(v,d).(!\, b_2 \twoheadrightarrow v \mid d \twoheadrightarrow b_1) \mid !\, x(w).\overline{b_2}\, w) \mid !\, b \twoheadrightarrow x \mid b_1 \twoheadrightarrow a) \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\parallel} b) & \sim_{\mathrm{C}} \quad (b,x) \\
(\nu b_1)\,((\nu c_1 b_2)\,(\llbracket \Delta \rrbracket^{\mathrm{F}}_{\parallel} c_1 \mid c_1(v,d).(!\, b_2 \twoheadrightarrow v \mid d \twoheadrightarrow b_1) \mid b_1 \twoheadrightarrow a \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\parallel} b_2) & \sim_{\mathrm{C}} \quad (b_1) \\
(\nu c_1 b_2)\,(\llbracket \Delta \rrbracket^{\mathrm{F}}_{\parallel} c_1 \mid c_1(v,d).(!\, b_2 \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\llbracket \Delta \rrbracket^{\mathrm{F}}_{\parallel} b_2) & \triangleq \llbracket \Delta\Delta \rrbracket^{\mathrm{F}}_{\parallel} a
\end{array}$$

This shows that the interpretation of $\Delta\Delta$ reduces (even using lazy reduction) in more than one step to an equivalent process, without creating output over a.

We can run inside the right-hand side of an application:

$$\begin{array}{ll}
\llbracket I(II) \rrbracket^{\mathrm{F}}_{\parallel} a \triangleq (\nu cb)\,(\llbracket I \rrbracket^{\mathrm{F}}_{\parallel} c \mid c(v,d).(!\, b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\parallel} b) & \equiv, \triangleq \\
(\nu cb)\,(\llbracket I \rrbracket^{\mathrm{F}}_{\parallel} c \mid c(v,d).(!\, b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid (\nu c_1 b_1)\,((\nu z b_2)\,(\llbracket z \rrbracket^{\mathrm{F}}_{\parallel} b_2 \mid \overline{c_1}\langle z,b_2\rangle) \mid & \\
\quad c_1(v,d).(!\, b_1 \twoheadrightarrow v \mid d \twoheadrightarrow b) \mid !\llbracket I \rrbracket^{\mathrm{F}}_{\parallel} b_1) \mid !\llbracket (\lambda z.z) I \rrbracket^{\mathrm{F}}_{\parallel} b) & \to_{\pi} (c_1) \\
(\nu cb)\,(\llbracket I \rrbracket^{\mathrm{F}}_{\parallel} c \mid c(v,d).(!\, b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid (\nu b_1)\,((\nu z b_2)\,(z(w).\overline{b_2}\, w \mid !\, b_1 \twoheadrightarrow z \mid b_2 \twoheadrightarrow b) \mid & \\
\quad (\nu y b_4)\,(\llbracket y \rrbracket^{\mathrm{F}}_{\parallel} b_4 \mid \overline{b_1}\langle y,b_4\rangle) \mid !\llbracket I \rrbracket^{\mathrm{F}}_{\parallel} b_1) \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\parallel} b) & \to_{\pi} (b_1,z,b_2) \\
(\nu cb)\,((\nu xb_5)\,(\llbracket x \rrbracket^{\mathrm{F}}_{\parallel} b_5 \mid \overline{c}\langle x,b_5\rangle) \mid c(v,d).(!\, b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid (\nu y b_4)\,(\llbracket y \rrbracket^{\mathrm{F}}_{\parallel} b_4 \mid \overline{b}\langle y,b_4\rangle) \mid & \\
\quad (\nu b_1 z)\,(!\, b_1 \twoheadrightarrow z \mid !\llbracket I \rrbracket^{\mathrm{F}}_{\parallel} b_1) \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\parallel} b) & \to_{\pi} (c) \\
(\nu b)\,((\nu xb_5)\,(x(w).\overline{b_5}\, w \mid !\, b \twoheadrightarrow x \mid b_5 \twoheadrightarrow a) \mid (\nu y b_4)\,(\llbracket y \rrbracket^{\mathrm{F}}_{\parallel} b_4 \mid \overline{b}\langle y,b_4\rangle) \mid & \\
\quad (\nu b_1 z)\,(!\, b_1 \twoheadrightarrow z \mid !\llbracket I \rrbracket^{\mathrm{F}}_{\parallel} b_1) \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\parallel} b) & \to_{\pi} (b,x,b_5) \\
(\nu y b_4)\,(\llbracket y \rrbracket^{\mathrm{F}}_{\parallel} b_4 \mid \overline{a}\langle y,b_4\rangle) \mid (\nu b_1 z)\,(!\, b_1 \twoheadrightarrow z \mid !\llbracket I \rrbracket^{\mathrm{F}}_{\parallel} b_1) \mid (\nu b x)\,(!\, b \twoheadrightarrow x \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\parallel} b) & \sim_{\mathrm{C}} \quad \llbracket I \rrbracket^{\mathrm{F}}_{\parallel} a
\end{array}$$

Notice that we created a copy of the replicated right-hand side first, and model more than just lazy or spine reduction. Since we reduced under the bound name b, this is not a reduction in $\to_{\mathrm{L}\pi}$; we can, however, also reduce $\llbracket I(II) \rrbracket^{\mathrm{F}}_{\parallel} a$ using lazy reduction:

$$\begin{array}{ll}
\llbracket I(II) \rrbracket^{\mathrm{F}}_{\parallel} a \triangleq (\nu cb)\,((\nu xb_1)\,(\llbracket x \rrbracket^{\mathrm{F}}_{\parallel} b_1 \mid \overline{c}\langle x,b_1\rangle) \mid c(v,d).(!\, b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\parallel} b) & \to_{\mathrm{L}\pi} (c) \\
(\nu b)\,((\nu xb_1)\,(\llbracket x \rrbracket^{\mathrm{F}}_{\parallel} b_1 \mid !\, b \twoheadrightarrow x \mid b_1 \twoheadrightarrow a \mid & \\
\quad (\nu cb_2)\,((\nu y b_3)\,(\llbracket y \rrbracket^{\mathrm{F}}_{\parallel} b_3 \mid \overline{c}\langle y,b_3\rangle) \mid c(v,d).(!\, b_2 \twoheadrightarrow v \mid d \twoheadrightarrow b) \mid !\llbracket I \rrbracket^{\mathrm{F}}_{\parallel} b_2) \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\parallel} b)) & \to_{\mathrm{L}\pi} (c) \\
(\nu b)\,((\nu xb_1)\,(x(w).\overline{b_1}\, w \mid b \twoheadrightarrow x \mid !\, b \twoheadrightarrow x \mid b_1 \twoheadrightarrow a \mid (\nu b_2)\,((\nu y b_3)\,(y(w).\overline{b_3}\, w \mid b_2 \twoheadrightarrow y \mid !\, b_2 \twoheadrightarrow y \mid & \\
\quad b_3 \twoheadrightarrow b) \mid (\nu z b_4)\,(\llbracket z \rrbracket^{\mathrm{F}}_{\parallel} b_4 \mid \overline{b_2}\langle z,b_4\rangle) \mid !\llbracket I \rrbracket^{\mathrm{F}}_{\parallel} b_2) \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\parallel} b)) & \to_{\mathrm{L}\pi} (b_2,y,b_3,b,x,b_1) \\
(\nu z b_4)\,(\llbracket z \rrbracket^{\mathrm{F}}_{\parallel} b_4 \mid \overline{a}\langle z,b_4\rangle) \mid (\nu b x b_1 b_2 y)\,(!\, b \twoheadrightarrow x \mid !\, b_2 \twoheadrightarrow y \mid !\llbracket I \rrbracket^{\mathrm{F}}_{\parallel} b_2 \mid !\llbracket II \rrbracket^{\mathrm{F}}_{\parallel} b) & \sim_{\mathrm{C}} \quad \llbracket I \rrbracket^{\mathrm{F}}_{\parallel} a
\end{array}$$

Notice that the term that gets discarded in the last step is mute.

In $\llbracket x(\Delta\Delta) \rrbracket^{\mathrm{F}}_{\parallel} a \triangleq (\nu cb)\,(\llbracket x \rrbracket^{\mathrm{F}}_{\parallel} c \mid c(v,d).(!\, b \twoheadrightarrow v \mid d \twoheadrightarrow a) \mid !\llbracket \Delta\Delta \rrbracket^{\mathrm{F}}_{\parallel} b)$, the output b of $\llbracket \Delta\Delta \rrbracket^{\mathrm{F}}_{\parallel} b$ is bound, so that process is mute, and $\llbracket x(\Delta\Delta) \rrbracket^{\mathrm{F}}_{\parallel} a$ is in lazy normal form.

4 Termination

Since through $\llbracket \cdot \rrbracket^{\mathrm{F}}_{\parallel} \cdot$ 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, as done in [**?**], this is relatively easy to fix by placing a guard on the replicated term, similar to what is done in Milner's encoding; see [**?**] for details. We cannot use that approach here, since it would invalidate our main representation result. We can, however, show termination for β-reduction, via 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 and is contextually equal to $\mathit{0}$.[8] 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 $(\nu x)\,(\mathbb{T}M_{\shortparallel}^{\mathrm{s}}a \mid !\,\mathbb{T}N_{\shortparallel}^{\mathrm{s}}x)$), and notice that $!\,\mathbb{T}N_{\shortparallel}^{\mathrm{s}}x \equiv \mathbb{T}N_{\shortparallel}^{\mathrm{s}}x \mid \cdots \mid \mathbb{T}N_{\shortparallel}^{\mathrm{s}}x \mid !\,\mathbb{T}N_{\shortparallel}^{\mathrm{s}}x$. Now in case x appears in M, only finitely many of the $\mathbb{T}N_{\shortparallel}^{\mathrm{s}}x$ will eventually communicate with the $x(w).\bar{b}\,w$ 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 $P \mid (\nu x)\,(!\,\mathbb{T}N_{\shortparallel}^{\mathrm{s}}x)$; 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 4.1 We can remove infinite computations that do not contribute to output:

$$\begin{array}{lll}
\mathbb{T}(\lambda x.I)(\Delta\Delta)_{\shortparallel}^{\mathrm{F}}a & \triangleq & \\
(\nu cb)\,((\nu x b_1)\,(\mathbb{T}I_{\shortparallel}^{\mathrm{F}}b_1 \mid \bar{c}\langle x,b_1\rangle) \mid c(v,d).(!\,b{\twoheadrightarrow}v \mid d{\twoheadrightarrow}a) \mid !\,\mathbb{T}\Delta\Delta_{\shortparallel}^{\mathrm{F}}b) & \rightarrow_{\pi}(c) & \\
(\nu b x b_1)\,(\mathbb{T}I_{\shortparallel}^{\mathrm{F}}b_1 \mid !\,b{\twoheadrightarrow}x \mid b_1{\twoheadrightarrow}a \mid !\,\mathbb{T}\Delta\Delta_{\shortparallel}^{\mathrm{F}}b) & \equiv & \\
(\nu b_1)\,(\mathbb{T}I_{\shortparallel}^{\mathrm{F}}b_1 \mid b_1{\twoheadrightarrow}a) \mid (\nu b x)\,(!\,b{\twoheadrightarrow}x \mid !\,\mathbb{T}\Delta\Delta_{\shortparallel}^{\mathrm{F}}b) & \sim_{\mathrm{C}} & \\
(\nu x)\,(\mathbb{T}I_{\shortparallel}^{\mathrm{F}}a \mid !\,\mathbb{T}\Delta\Delta_{\shortparallel}^{\mathrm{F}}x) & \equiv & \\
\mathbb{T}I_{\shortparallel}^{\mathrm{F}}a \mid (\nu x)\,(!\,\mathbb{T}\Delta\Delta_{\shortparallel}^{\mathrm{F}}x) & \sim_{\mathrm{C}} & \mathbb{T}I_{\shortparallel}^{\mathrm{F}}a
\end{array}$$

In fact, 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 equivalent to the process with these parts removed.

Theorem 4.2 (Termination) *$M\!\downarrow$, then $\mathbb{T}M_{\shortparallel}^{\mathrm{F}}a\!\downarrow_{\mathrm{L}\pi}$.*

Notice that the normal form of $\mathbb{T}M_{\shortparallel}^{\mathrm{F}}a$ need not be the interpretation of the normal form of M, but that these are contextually equivalent; this is also the case for Milner's interpretation. However, when interpreting *explicit spine* reduction we never need to reduce inside the argument of an application or inside a substitution, so we can show:

Theorem 4.3 *$M \rightarrow_{\mathbf{xs}} N$, and N is in normal form, then $\mathbb{T}N_{\shortparallel}^{\mathrm{F}}a$ is in lazy normal form.*

This is a stronger result than the one obtained in [7] (Thm. 4.6 here) where it is shown with respect to the *explicit* spine reduction. Notice that lazy π-reduction is used in these results.

Moreover, the notion of type assignment we defined above catches this: since $\mathbb{T}N_{\shortparallel}^{\mathrm{F}}x$ has at most only one visible output (which is x), the process $(\nu x)\,(!\,\mathbb{T}N_{\shortparallel}^{\mathrm{F}}x)$ has *no* visible output, and can therefore be typed by $(\nu x)\,(!\,\mathbb{T}N_{\shortparallel}^{\mathrm{F}}x) : \Gamma \vdash_{\pi}^{io}$, *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 [27]. 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.

[8] Alternatively, we could replace the congruence rule $!\,P \equiv P \mid !\,P$ by $(\nu a)\,(a(y).R \mid Q \mid !\,P) \equiv (\nu a)\,(!\,a(y).R \mid P) \mid (\nu a)\,(Q \mid !\,P)$ only if a is an output in P.

We have found a new, simple and intuitive interpretation of λ-terms in π that respects explicit reduction, full step-by-step β-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 $x(w).\overline{a}\,w$, $b(v,d).(!\,a \rightarrow v \mid d \rightarrow c)$, 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 $x(w).\overline{a_1}\,w \mid \cdots \mid x(w).\overline{a_n}\,w$ inside $[\![M]\!]^{\mathrm{F}}_{\parallel} a$; it seems plausible to use *broadcast communication* [18] – 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.

Also, it seems that the interpretations generate a notion of proof net in π; in fact, it seems promising to encode *linear* $\mathcal{X}$ ($*\mathcal{X}$, defined in [29]), a calculus with explicit contraction and weakening, since there reduction can generate non-connected terms, a feature we observe also in our interpretations; we hope to then strengthen our results by, perhaps, using strong bi-simulation.

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. Proofs as Processes. *Theoretical Computer Science*, 135(1):5–9, 1994.

[5] 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.

[6] S. van Bakel and M. Fernández. Normalisation, Approximation, and Semantics for Combinator Systems. *Theoretical Computer Science*, 290:975–1019, 2003.

[7] 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.

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

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

[10] 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.

[11] P. Bruscoli. A Purely Logical Account of Sequentiality in Proof Search. In P.J. Stuckey, editor, *18th International Conference on Logic Programming (ICLP '02), Copenhagen, Denmark*, volume 2401 of *Lecture Notes in Computer Science*, pages 302–316. Springer Verlag, 2002.

[12] P. Bruscoli and A. Guglielmi. A Linear Logic Programming Language with Parallel and Sequential Conjunction. In M. Alpuente and M.I. Sessa, editors, *Joint Conference on Declarative Programming (GULP-PRODE'95), Marina di Vietri, Italy*, pages 409–420, 1995.
[13] A. Church. A Note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1(1):40–41, 1936.
[14] M. Cimini, C. Sacerdoti Coen, and D. Sangiorgi. $\overline{\lambda}\mu\tilde{\mu}$ calculus, π-calculus, and abstract machines. 2009.
[15] M.D. Clinger. *Foundations of Actor Semantics*. PhD thesis, MIT Artificial Intelligence Laboratory, 1981. Technical Report 633.
[16] H.B. Curry. Grundlagen der Kombinatorischen Logik. *American Journal of Mathematics*, 52:509–536, 789–834, 1930.
[17] G. Gentzen. Untersuchungen über das Logische Schliessen. *Mathematische Zeitschrift*, 39:176–210 and 405–431, 1935.
[18] M. Hennessy and J. Rathke. Bisimulations for a Calculus of Broadcasting Systems. *Theoretical Computer Science*, 200(1-2):225–260, 1998.
[19] 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.
[20] K. Honda and N. Yoshida. On the reduction-based process semantics. *Theoretical Computer Science*, 151:437–486, 1995.
[21] K. Honda, N. Yoshida, and M. Berger. Control in the π-Calculus. In *Proceedings of Fourth ACM-SIGPLAN Continuation Workshop* (CW'04), 2004.
[22] S. Maffeis and I.C.C. Phillips. On the Computational Strength of Pure Ambient Calculi. In *Proceedings of Express'03*, volume 91 of *Electronic Notes in Theoretical Computer Science*, 2004.
[23] R. Milner. Functions as Processes. *Mathematical Structures in Computer Science*, 2(2):269–310, 1992.
[24] 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.
[25] B.C. Pierce and D. Sangiorgi. Typing and Subtyping for Mobile Processes. *Mathematical Structures in Computer Science*, 6(5):409–453, 1996.
[26] D. Sangiorgi. Lazy functions and mobile processes. Rapport de Recherche 2515, INRIA, Sophia-Antipolis, France, 1995.
[27] D. Sangiorgi and D. Walker. *The Pi-Calculus*. Cambridge University Press, 2001.
[28] H. Thielecke. *Categorical Structure of Continuation Passing Style*. PhD thesis, University of Edinburgh, 1997. LFCS technical report ECS-LFCS-97-376.
[29] D. Žunić. *Computing with Sequents and Diagrams in Classical Logic - Calculi* ${}^{*}\mathcal{X}$, ${}^{d}\mathcal{X}$, *and* ${}^{©}\mathcal{X}$. PhD thesis, École Normale Supèrieure de Lyon, 2007.