CLASSICAL CUT-ELIMINATION IN THE π-CALCULUS

STEFFEN VAN BAKEL, LUCA CARDELLI, AND MARIA GRAZIA VIGLIOTTI

Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2BZ, UK
e-mail address: svb@doc.ic.ac.uk

Microsoft Research Cambridge, 7 J J Thomson Avenue, Cambridge, CB3 0FB, UK
e-mail address: luca@microsoft.com

Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2BZ, UK
e-mail address: mgv98@doc.ic.ac.uk

ABSTRACT. We study the π-calculus, enriched with pairing, and define a notion of type assignment that uses the type constructor $\rightarrow$. We encode the terms of the calculus $\mathcal{X}$ into this variant of π, and show that all reduction and assignable types are preserved. Since $\mathcal{X}$ enjoys the Curry-Howard isomorphism for Gentzen's calculus LK, this implies that all proofs in LK have a representation in π, and *cut*-elimination is simulated by π's synchronisation of processes. We then enrich the logic with the connector $\neg$, and show that this also can be represented in π.

INTRODUCTION

In this paper we present three encodings of proofs of Gentzen's (implicative) LK [24] into the π-calculus [36] that respect *cut*-elimination, and define a new notion of type assignment for π so that processes will become witnesses for the provable formulae. These encodings of classical logic into π-calculus are attained by using the intuition of the calculus $\mathcal{X}$, which gives a computational meaning to LK (a first version of this calculus was proposed in [45, 47, 46]; the implicative fragment of $\mathcal{X}$ was studied in [10]).

$\mathcal{X}$ enjoys the Curry-Howard isomorphism for LK, which it achieves by inhabiting the inference rules with term information, constructing witnesses for derivable sequents. Terms in $\mathcal{X}$ have multiple named inputs and multiple named outputs, that are collectively called *connectors*. Reduction in $\mathcal{X}$ is expressed via a set of rewrite rules that represent/correspond to *cut*-elimination in LK; reducing a term using these rules eventually leads to renaming of connectors and gives computational meaning to classical (sequent) proof reduction. It is well known that *cut*-elimination in LK is not confluent, and, since $\mathcal{X}$ is Curry-Howard for LK and its reduction respects *cut*-elimination, neither is reduction in $\mathcal{X}$.

These two features –non-confluence and reduction as connection of terms via the exchange of names– inspired us to consider the π-calculus as an alternative computational model for *cut*-elimination and proofs in LK. The relation between process calculi and classical logic is an interesting and very promising area of research (similar attempts were made in the context of natural

1998 ACM Subject Classification: F.1.2, F.4.1, D.3.1.
Key words and phrases: classical logic, sequent calculus, pi calculus, encoding, type assignment.

deduction [33] and linear logic [5, 15, 19, 18]). Our aim is to widen further the path to practical application of classical logic in computation by providing expressive interpretations of classical logic into process algebra, that fully exploit the non-determinism of both LK and π.

The aim of this paper is to link LK and π via $\mathcal{X}$; the main achievements are:

- encodings of $\mathcal{X}$ into π are defined that preserve the operational semantics; one that respects head-reduction, and the other two that respect reduction in full – to achieve these results, reduction in π is generalised by adding pairing [2];
- we define a non-standard notion of type assignment for π (types do not contain channel information) that encompasses implication;
- the encoding preserves assignable types, effectively showing that all proofs in LK have a representation in π;
- in addition to [10], we treat the connective $\neg$ as well.

Classical sequents. The *sequent calculus* LK, introduced by Gentzen in [24], is a logical system in which the rules only introduce connectives (but on either side of a sequent), in contrast to *natural deduction* (also introduced in [24]) which uses rules that introduce or eliminate connectives in the logical formulae. Natural deduction normally derives statements with a single conclusion, whereas LK allows for multiple conclusions, deriving sequents of the form $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$, where $A_1, \ldots, A_n$ is to be understood as $A_1 \wedge \ldots \wedge A_n$ and $B_1, \ldots, B_m$ is to be understood as $B_1 \vee \ldots \vee B_m$. The version G_3 [34], with implicit weakening and contraction, of Implicative LK has four rules: *axiom*, *left introduction* of the arrow, *right introduction*, and *cut*.

$$(Ax) : \frac{}{\Gamma, A \vdash A, \Delta} \qquad (\Rightarrow L) : \frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}$$

$$(\Rightarrow R) : \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta} \qquad (cut) : \frac{\Gamma \vdash A, \Delta \quad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}$$

Since LK has only introduction rules, the only way to eliminate a connective is to eliminate the whole formula in which it appears via an application of the (cut)-rule. Gentzen defined a procedure that eliminates all applications of the (cut)-rule from a proof of a sequent using an innermost strategy, generating a proof in *normal form* of the same sequent, *i.e.*, without a *cut*. This procedure is defined via local reductions of the proof-tree, which has –with some discrepancies– the flavour of term rewriting [35] or the evaluation of explicit substitutions [17, 1]. Indeed, the typing rule of an explicit substitution, say in $\lambda\mathbf{x}$ [16], is nothing but a variant of the (cut)-rule, and a lot of work has been done to better understand the connection between explicit substitutions and local *cut*-reduction procedures.

The principle of $\mathcal{X}$. The calculus $\mathcal{X}$ achieves a Curry-Howard isomorphism, first discovered for Combinatory Logic [23], for the proofs in LK by constructing *witnesses* for derivable sequents. In establishing the isomorphism for $\mathcal{X}$, similar to calculi like $\lambda\mu$ [38] and $\overline{\lambda}\mu\tilde{\mu}$ [22], Roman names are attached to formulae in the left context, and Greek names for those on the right, and syntactic structure is associated to the rules. Names on the left can be seen as inputs to the term, and names to the right as outputs; since multiple formulae can appear on both sides, this implies that a term can not only have more than one input, but also more than one output. There are two kinds of names (connectors) in $\mathcal{X}$: *sockets* (inputs, with Roman names, that correspond to values) and *plugs* (outputs, with Greek names, that correspond to continuations), that correspond to *variables* and *co-variables*, respectively, in [48], or, alternatively, to Parigot's λ and μ-variables (see also [22]).

In the construction of the witness, when in applying a rule a premise or conclusion disappears from the sequent, the corresponding name gets bound in the term that is constructed, and when a premise or conclusion gets created, a different free (often, but not necessarily, new) name is associated to it. For example, in the creation of the term for right-introduction of the arrow

$$\frac{P \mathrel{:\cdot} \Gamma, x{:}A \vdash_{\mathcal{X}} \alpha{:}B, \Delta}{\widehat{x}P\widehat{\alpha}\cdot\beta \mathrel{:\cdot} \Gamma \vdash_{\mathcal{X}} \beta{:}A\to B, \Delta}$$

the input x and the output α are bound, and β is free. This case is interesting in that it highlights a special feature of $\mathcal{X}$, not found in other calculi. In (applicative) calculi related to natural deduction, like the λ-calculus, only inputs are named, and the linking to a term that will be inserted is done via λ-abstraction and application. The output (*i.e.* result) on the other hand is anonymous; where a term 'moves to' carries a name via a variable that acts as a pointer to the positions where the term is to be inserted, but where it comes from is not mentioned, and left implicit. Since in $\mathcal{X}$ a term P can have many inputs and outputs, it is unsound to consider P a function *per se*; however, fixing *one* input x and *one* output α, we can see P as a function 'from x to α'. We make this limited view of P available via the output β, thereby *exporting* via β that 'P can be used as a function from x to α'. The types given to the connectors confirm this view.

Gentzen's proof reductions by *cut*-elimination become the fundamental principle of computation in $\mathcal{X}$. *Cuts* in proofs are witnessed by $P\widehat{\alpha} \dagger \widehat{x}Q$ (called the *cut* of P and Q via α and x), and the reduction rules specify how to remove them: a term is in normal form if and only if it has no sub-term of this shape. The intuition behind reduction is: the *cut* $P\widehat{\alpha} \dagger \widehat{x}Q$ expresses the intention to connect all αs in P and xs in Q, and reduction will realise this by either connecting all αs to all xs (if x does not exist in Q, P will disappear), or all xs to all αs (if α does not exist in P, Q will disappear). Since *cut*-elimination in LK is not confluent, neither is reduction in $\mathcal{X}$; for example, as suggested above, when P does not contain α and Q does not contain x, reducing $P\widehat{\alpha} \dagger \widehat{x}Q$ can lead to both P and Q, two different terms.

Reduction in $\mathcal{X}$ boils down to *renaming*: since the calculus is substitution-free, during reduction terms are re-organised, creating terms that are similar, but with different connector names inside.

Capturing $\mathcal{X}$ in π. $\mathcal{X}$'s notion of multiple inputs and outputs is also found in π, and was the original inspiration for our research. The aim of this work is to find a simple and intuitive encoding of LK-proofs in π, and to devise a notion of type assignment for π so that the types in $\mathcal{X}$ are preserved in π. In this precise sense we view processes in π as giving an alternative computational meaning to proofs in classical logic. To achieve this goal, we made full use of the view of $\mathcal{X}$-terms sketched above. Clearly this implies that we had to define a notion of type assignment that uses the type constructor $\to$ for π; we managed this without having to linearise the calculus as done in [33], and this is one of the contributions of this paper.

Although the calculi $\mathcal{X}$ and π are, of course, essentially different, the similarities go beyond the correspondence of inputs and output between terms in $\mathcal{X}$ and processes in π. Like $\mathcal{X}$, π is application free, and substitution only takes place on *channel names*, similar to the renaming feature of $\mathcal{X}$, so *cut*-elimination is similar to synchronisation.

As discussed above, when creating a witness for $(\Rightarrow R)$ (the term $\widehat{x}P\widehat{\alpha}\cdot\beta$, called an *export*), the exported interface of P is the functionality of 'receiving on x, sending on α', which is made available on β. When encoding this behaviour in π, we are faced with a problem. It is clearly not sufficient to limit communication to the exchange of single names, since then we would have to separately send x and α, breaking perhaps the exported functionality, and certainly disabling the possibility of assigning arrow types. We overcome this problem by sending out a pair of names, as in $\overline{a}\langle\langle v,\delta\rangle\rangle$. Similarly, when interpreting a witness for $(\Rightarrow L)$ (the term $P\widehat{\alpha}\,[x]\,\widehat{y}Q$, called an *import*), the term that is to be connected to x is ideally a function whose input will be connected to α, and its output to y. This means that we need to receive a pair of names over x, as in $x(\langle v,\delta\rangle).P$.

A *cut* $P\widehat{\alpha} \dagger \widehat{x}Q$ in $\mathcal{X}$ expresses two terms that need to be connected via α and x. If we model P and Q in π, then we obtain one process sending on α, and one receiving on x, and we need to link these via $\alpha(w).\overline{x}\langle w\rangle$. Since each output on α in P takes place only once, and Q might want to receive in more than one x, we need to replicate the sending; likewise, since each input x in Q takes place only once, and P might have more than one send operation on α, Q needs to be replicated.

Related work. The relation between *logic* and *computation* hinges around the Curry-Howard isomorphism (sometimes also attributed to de Bruijn), which expresses the fact that, for certain calculi with a notion of types, there exists a corresponding logic such that it becomes possible to associate terms with proofs, linking the term's type to the proposition shown by the proof, and proof contractions become term reductions (or computations). This phenomenon was first discovered for Combinatory Logic [23], and played an important part in de Bruijn's Automath[1].

Before Herbelin's PhD [29] and Urban's PhD [45], the study of the relation between computation, programming languages and logic has concentrated mainly on *natural deduction systems* (of course, exceptions exist [25, 26]). In fact, these carry the predicate '*natural*' deservedly; in comparison with, for example, *sequent style systems*, natural deduction systems are easy to understand and reason about. This holds most strongly in the context of *non-classical* logics; for example, the Curry-Howard relation between *Intuitionistic Logic* and the *Lambda Calculus* with types – of which the basic system is formulated by

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

– is well studied and understood, and has resulted in a vast and well-investigated area of research, resulting in, amongst others, functional programming languages and much further to system F [27] and the Calculus of Constructions [21]. Abramsky [4, 5] has studied correspondence between multiplicative linear logic and processes, and later moved to the context of game semantics [6]. In fact, all the calculi are *applicative* in that abstraction and application (corresponding to arrow introduction and elimination) are the main constructors in the syntax.

The link between Classical Logic and continuations and control was first established for the λ_C-Calculus [28] (where C stands for Felleisen's C operator). Not much later, Parigot presented his $\lambda\mu$-calculus [38], an approach for representing classical proofs via a natural deduction system in which there is one main conclusion that is being manipulated, and possibly several alternative ones; the corresponding logic is one with *focus*. The $\lambda\mu$-calculus is presented as an extension of the λ-calculus, by extending the syntax with two new constructs that act as witness to the rules that deal with *conflict* ($\perp$):

$$\frac{\Gamma \vdash_{\lambda\mu} M:A \mid \alpha{:}A,\Delta}{\Gamma \vdash_{\lambda\mu} [\alpha]M:\perp \mid \alpha{:}A,\Delta} \qquad \frac{\Gamma \vdash_{\lambda\mu} M:\perp \mid \alpha{:}A,\Delta}{\Gamma \vdash_{\lambda\mu} \mu\alpha.M:A \mid \Delta}$$

[1] `http://www.win.tue.nl/automath`

It uses two disjoint sets of variables (Roman letters and Greek letters). The sequents typing terms are of the form $\Gamma \vdash A \mid \Delta$, marking the conclusion A as *active*.

The introduction-elimination approach is easy to understand and convenient to use, but is also rather restrictive: for example, the handling of negation is not as nicely balanced, as is the treatment of contradiction (for a detailed discussion, see [42]). This imbalance can be observed in the $\lambda\mu$-calculus: adding $\perp$ as pseudo-type (only negation, or $A \rightarrow \perp$, is expressed; $\perp \rightarrow A$ is not a type), the $\lambda\mu$-calculus corresponds to *minimal classical logic* [7].

Herbelin has studied the calculus $\overline{\lambda}\mu\tilde{\mu}$ as a non-applicative extension of $\lambda\mu$, which gives a fine-grained account of manipulation of sequents [29, 22, 30]. The relation between call-by-name and call-by-value in the fragment of LK with negation and conjunction is studied in the Dual Calculus [48]; as in calculi like $\lambda\mu$ and $\overline{\lambda}\mu\tilde{\mu}$, that calculus considers a logic with *active* formulae, so these calculi do not achieve a direct Curry-Howard isomorphism with LK. The relation between $\mathcal{X}$ and $\overline{\lambda}\mu\tilde{\mu}$ has been investigated in [9, 10]; there it was shown that it is straightforward to map $\overline{\lambda}\mu\tilde{\mu}$-terms into $\mathcal{X}$ whilst preserving reduction, but that it is not possible to do the converse.

The π-calculus is equipped with a rich type theory [41]: from the basic type system for counting the arity of channels [39] to sophisticated linear types in [33], which studies a relation between Call-by-Value $\lambda\mu$ and a linear π-calculus. Linearisation is used to be able to achieve processes that are functions, by allowing output over one channel name only. Moreover, the encoding presented in [33] is type dependent, in that, for each term, there are different π-processes assigned, depending on the original type; this makes the encoding quite cumbersome. By contrast, our encoding is very simple and intuitive by interpreting the *cut* operationally as a communication. The idea of giving a computational interpretation of the *cut* as a communication primitive is also used in [5] and [15]. In both these papers, only a small fragment of Linear Logic was considered, and the encoding between proofs and π-calculus was left rather implicit.

The type system presented in this paper differs quite drastically from the standard type system presented in [41] in that our types contain no channel information: here input and output channels essentially have the type of the data they are sending or receiving, and are separated by the type system by putting all inputs with their types on the left of the sequent, and the outputs on the right. In our paper, types give a logical view to the π-calculus rather than an abstract specification on how channels should behave.

A result similar to ours has appeared as [20], but for the fact that there a relation is established between the $\overline{\lambda}\mu\tilde{\mu}$-calculus and the π-calculus; $\overline{\lambda}\mu\tilde{\mu}$ has a Curry-Howard relation with a version of LK with *activated formulae*, as in Parigot's $\lambda\mu$, so does not directly represent LK. The interpretation as defined in [20] strongly depends on recursion, is not compositional, and preserves only outermost reduction, not the (larger) notion of head-reduction we encode with $[\![\cdot]\!]_{\mathrm{S}}$; it does follow the reduction in $\overline{\lambda}\mu\tilde{\mu}$ closely, though. Also, since in that approach all communication takes place via channels named λ, μ and $\tilde{\mu}$, it is not immediately clear that a natural notion of type assignment exists for π so that also type assignment is preserved.

Overview of this paper. In Section 1, we briefly repeat the definitions of (implicative) $\mathcal{X}$, followed by the notion of type assignment which establishes the Curry-Howard isomorphism. In Section 2, we show how to rewrite the $\mathcal{X}$-terms, and show the relation with LK's *cut*-elimination. The π-calculus with pairing is presented in Section 3. Section 4 defines the notion of head-reduction in $\mathcal{X}$, which is encoded into π via $[\![\cdot]\!]_{\mathrm{S}}$; in Section 5 we will modify this encoding to full represent $\mathcal{X}$'s reduction, via the encodings $[\![\cdot]\!]_{\mathrm{F}}$ and $[\![\cdot]\!]_{\mathrm{R}}$.

In Section 6, we define a notion of type assignment for the π-calculus. Then, in Section 7 we look at how to represent the other connectives in $\mathcal{X}$, and study the relation between these representations and reduction. We conclude by extending the syntax of names in π to elegantly represent the other connectives directly in π.

In [8], we first presented our results on the encoding of $\mathcal{X}$-terms in to the π-calculus; that paper also presented the notion of type assignment as defined here, as well as a proof that type assignment is preserved by the encoding. We repeat these results here, with all details of the proofs; however, here we define a notion of head-reduction '$\rightarrow_{\mathrm{H}}$' for $\mathcal{X}$, and show that the encoding $\llbracket \cdot \rrbracket_{\mathrm{S}}$ respects $\rightarrow_{\mathrm{H}}$; we also add the encodings $\llbracket \cdot \rrbracket_{\mathrm{F}}$ and $\llbracket \cdot \rrbracket_{\mathrm{R}}$ and show that these are faithful with respect to $\mathcal{X}$'s full reduction.

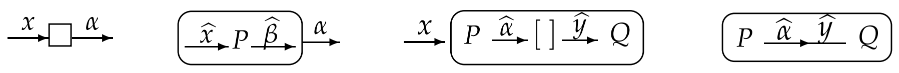
1. The calculus $\mathcal{X}$

In this section and the next we will give the definition of the $\mathcal{X}$-calculus which has been proven to be a fine-grained implementation model for various well-known calculi [9], like the λ-calculus [14], $\lambda\mu$, and $\overline{\lambda}\mu\tilde{\mu}$. As discussed in the introduction, the calculus $\mathcal{X}$ is inspired by the sequent calculus; the system we will consider in this section has only implication, no structural rules and a changed axiom; we will consider the other connectives in Section 7. $\mathcal{X}$ features two separate categories of 'connectors', *plugs* and *sockets*, that act as input and output channels, and is defined without any notion of substitution or application.

Definition 1.1 (Syntax)**.** The terms of the $\mathcal{X}$-calculus are defined by the following syntax, where the Roman characters x, y range over the infinite set of *sockets*, and the Greek characters α, β over the infinite set of *plugs*.

$$\begin{array}{rcll} P, Q & ::= & \langle x{\cdot}\alpha\rangle & \textit{capsule} \\ & \mid & \widehat{y}P\widehat{\beta}\cdot\alpha & \textit{export} \\ & \mid & P\widehat{\alpha}\,[x]\,\widehat{y}Q & \textit{import} \\ & \mid & P\widehat{\alpha}\dagger\widehat{y}Q & \textit{cut} \end{array}$$

We can represent these terms via the following diagrams (given just as a visual aid).



As an aid to intuition, ignoring the explicitly named outputs, we can see these terms with the view of the λ-calculus: the *capsule* $\langle x{\cdot}\alpha\rangle$ can then be seen as the variable x, the *export* $\widehat{x}P\widehat{\alpha}\cdot\beta$ as the abstraction $\lambda x.P$, the *import* $P\widehat{\alpha}\,[x]\,\widehat{y}Q$ as the term $xPQ_1\cdots Q_n$ (where Q is seen as a context, acting as a stack of terms $Q_1,\ldots,Q_n$), and the *cut* $P\widehat{\alpha}\dagger\widehat{x}Q$ as the substitution $Q\,\langle x := P\rangle$.

The $\hat{\ }$ symbolises that the socket or plug underneath is bound in the term. The notion of bound and free connector is defined as usual, and we will identify terms that only differ in the names of bound connectors, as usual.

Definition 1.2. The *free sockets* and *free plugs* in a net are defined by:

$$\begin{array}{lcllcl} \mathit{fs}\,(\langle x{\cdot}\alpha\rangle) & = & \{x\} & \mathit{fp}(\langle x{\cdot}\alpha\rangle) & = & \{\alpha\} \\ \mathit{fs}\,(\widehat{x}P\widehat{\alpha}\cdot\beta) & = & \mathit{fs}\,(P)\backslash\{x\} & \mathit{fp}(\widehat{x}P\widehat{\alpha}\cdot\beta) & = & (\mathit{fp}(P)\backslash\{\alpha\})\cup\{\beta\} \\ \mathit{fs}\,(P\widehat{\alpha}\,[y]\,\widehat{x}Q) & = & \mathit{fs}\,(P)\cup\{y\}\cup(\mathit{fs}\,(Q)\backslash\{x\}) & \mathit{fp}(P\widehat{\alpha}\,[y]\,\widehat{x}Q) & = & (\mathit{fp}(P)\backslash\{\alpha\})\cup\mathit{fp}(Q) \\ \mathit{fs}\,(P\widehat{\alpha}\dagger\widehat{x}Q) & = & \mathit{fs}\,(P)\cup(\mathit{fs}\,(Q)\backslash\{x\}) & \mathit{fp}(P\widehat{\alpha}\dagger\widehat{x}Q) & = & (\mathit{fp}(P)\backslash\{\alpha\})\cup\mathit{fp}(Q) \end{array}$$

A socket x or plug α occurring in P which is not free is called *bound*, written $x \in bs(P)$ and $\alpha \in bp(P)$. We will write $x \notin fs(P, Q)$ for $x \notin fs(P)$ & $x \notin fs(Q)$.

The set of *free connectors* of P is defined by: $fc(P) = fs(P) \cup fp(P)$.

We accept Barendregt's convention on names, which states that no name can occur both free *and* bound in a context; α-conversion is supposed to take place silently, whenever necessary. We will consider also, for example, x bound in $P[y/x]$ and $P \mathrel{:\cdot} \Gamma, x{:}A \vdash_{\mathcal{X}} \Delta$.

We first define types and contexts.

Definition 1.3 (Types and Contexts)**.** (1) The set of types is defined by the grammar:

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

where φ is a basic type of which there are infinitely many[2].

(2) A *context of sockets* Γ is a mapping from sockets to types, denoted as a finite set of *statements* $x{:}A$, such that the *subject* of the statements (x) are distinct. We write Γ_1, Γ_2 for the *compatible* union of Γ_1 and Γ_2 (if Γ_1 contains $x{:}A_1$ and Γ_2 contains $x{:}A_2$ then $A_1 = A_2$), and write $\Gamma, x{:}A$ for $\Gamma, \{x{:}A\}$. So, when writing a context as $\Gamma, x{:}A$, this implies that $x{:}A \in \Gamma$, or Γ is not defined on x.

(3) Contexts of *plugs* Δ, and the notions Δ_1, Δ_2 and $\alpha{:}A, \Delta$ are defined in a similar way.

The notion of type assignment on $\mathcal{X}$ that we present in this section is the basic implicative system for Classical Logic (Gentzen's system LK) as described above. The Curry-Howard property is easily achieved by erasing all term-information. When building witnesses for proofs, propositions receive names; those that appear in the left part of a sequent are named with Roman characters like x, y, z, etc, and those that appear in the right part of a sequent are named with Greek characters like α, β, γ, etc. When in applying a rule a formula disappears from the sequent, the corresponding connector will get bound in the term that is constructed, and when a formula gets created, a new connector will be associated to it.

Definition 1.4 (Typing for $\mathcal{X}$)**.** (1) *Type judgements* are expressed via a ternary relation $P \mathrel{:\cdot} \Gamma \vdash \Delta$, where Γ is a context of *sockets* and Δ is a context of *plugs*, and P is a term. We say that P is the *witness* of this judgement.

(2) *Type assignment for* $\mathcal{X}$ is defined by the following rules:

$$(cap): \frac{}{\langle y{\cdot}\alpha\rangle \mathrel{:\cdot} \Gamma, y{:}A \vdash \alpha{:}A, \Delta} \qquad (cut): \frac{P \mathrel{:\cdot} \Gamma \vdash \alpha{:}A, \Delta \quad Q \mathrel{:\cdot} \Gamma, x{:}A \vdash \Delta}{P\widehat{\alpha} \dagger \widehat{x}Q \mathrel{:\cdot} \Gamma \vdash \Delta}$$

$$(exp): \frac{P \mathrel{:\cdot} \Gamma, x{:}A \vdash \alpha{:}B, \Delta}{\widehat{x}P\widehat{\alpha}{\cdot}\beta \mathrel{:\cdot} \Gamma \vdash \beta{:}A{\to}B, \Delta} \qquad (imp): \frac{P \mathrel{:\cdot} \Gamma \vdash \alpha{:}A, \Delta \quad Q \mathrel{:\cdot} \Gamma, x{:}B \vdash \Delta}{P\widehat{\alpha}\,[y]\,\widehat{x}Q \mathrel{:\cdot} \Gamma, y{:}A{\to}B \vdash \Delta}$$

We write $P \mathrel{:\cdot} \Gamma \vdash_{\mathcal{X}} \Delta$ if there exists a derivation using these rules that has this judgement in the bottom line, and write $\mathcal{D} :: P \mathrel{:\cdot} \Gamma \vdash_{\mathcal{X}} \Delta$ if we want to name that derivation.

As in $\overline{\lambda}\mu\tilde{\mu}$, the term that inhabits left-introduction of the arrow, $Q\widehat{\alpha}\,[z]\,\widehat{v}R$, can be seen as a context with a hole (which in our case carries the name z), or as a list with Q at the head and R at the tail[3].

Notice that Γ and Δ carry the types of the free connectors in P, as unordered sets. There is no notion of type for P itself, instead the derivable statement shows how P is connectable.

[2]These types are normally known as *simple* (or *Curry*) types.

[3]In $\overline{\lambda}\mu\tilde{\mu}$, $(\Rightarrow L)$ is inhabited by $v \cdot e$, with v a term, and e a context, and, in fact, $\llbracket v{\cdot}e \rrbracket_x \triangleq \llbracket v \rrbracket_\alpha \widehat{\alpha}\,[x]\,\widehat{y} \llbracket e \rrbracket_y$, where $\llbracket \cdot \rrbracket_.$ is the interpretation of $\overline{\lambda}\mu\tilde{\mu}$ terms into $\mathcal{X}$; for details, see [10].

Example 1.5 (A proof of Peirce's Law)**.** The following is a proof for Peirce's Law in LK:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A, B}\,(Ax)}{\vdash A{\Rightarrow}B, A}\,(\Rightarrow R) \qquad \dfrac{}{A \vdash A}\,(Ax)}{(A{\Rightarrow}B){\Rightarrow}A \vdash A}\,(\Rightarrow L)}{\vdash ((A{\Rightarrow}B){\Rightarrow}A){\Rightarrow}A}\,(\Rightarrow R)$$

Inhabiting this proof in $\mathcal{X}$ gives the derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle y{\cdot}\delta\rangle \because y{:}A \vdash_{\mathcal{X}} \delta{:}A, \eta{:}B}\,(cap)}{\widehat{y}\langle y{\cdot}\delta\rangle\widehat{\eta}{\cdot}\alpha \because \ \vdash_{\mathcal{X}} \alpha{:}A{\to}B, \delta{:}A}\,(exp) \qquad \dfrac{}{\langle w{\cdot}\delta\rangle \because w{:}A \vdash_{\mathcal{X}} \delta{:}A}\,(cap)}{(\widehat{y}\langle y{\cdot}\delta\rangle\widehat{\eta}{\cdot}\alpha)\widehat{\alpha}\,[z]\,\widehat{v}\langle v{\cdot}\delta\rangle \because z{:}(A{\to}B){\to}A \vdash_{\mathcal{X}} \delta{:}A}\,(imp)}{\widehat{z}((\widehat{y}\langle y{\cdot}\delta\rangle\widehat{\eta}{\cdot}\alpha)\widehat{\alpha}\,[z]\,\widehat{v}\langle v{\cdot}\delta\rangle)\widehat{\delta}{\cdot}\gamma \because \ \vdash_{\mathcal{X}} \gamma{:}((A{\to}B){\to}A){\to}A}\,(exp)$$

2. Reduction on $\mathcal{X}$

The reduction rules for the calculus $\mathcal{X}$ are directly inspired by the *cut*-elimination rules in LK. For example, since

$$\dfrac{\dfrac{\dfrac{\mathcal{D}_1}{\Gamma, A \vdash_{\text{LK}} B, \Delta}}{\Gamma \vdash_{\text{LK}} A{\to}B, \Delta}\,(\to R) \qquad \dfrac{\dfrac{\mathcal{D}_2}{\Gamma \vdash_{\text{LK}} A, \Delta} \quad \dfrac{\mathcal{D}_3}{\Gamma, B \vdash_{\text{LK}} \Delta}}{\Gamma, A{\to}B \vdash_{\text{LK}} \Delta}\,(\to L)}{\Gamma \vdash_{\text{LK}} \Delta}\,(cut)$$

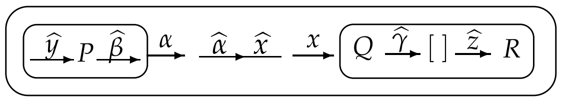
contracts to both

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{D}_2}{\Gamma \vdash_{\text{LK}} A, \Delta}}{\Gamma \vdash_{\text{LK}} A, B, \Delta}\,(W) \quad \dfrac{\mathcal{D}_1}{\Gamma, A \vdash_{\text{LK}} B, \Delta}}{\Gamma \vdash_{\text{LK}} B, \Delta}\,(cut) \quad \dfrac{\mathcal{D}_3}{\Gamma, B \vdash_{\text{LK}} \Delta}}{\Gamma \vdash_{\text{LK}} \Delta}\,(cut)$$

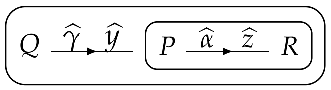
and

$$\dfrac{\dfrac{\mathcal{D}_2}{\Gamma \vdash_{\text{LK}} A, \Delta} \quad \dfrac{\dfrac{\mathcal{D}_1}{\Gamma, A \vdash_{\text{LK}} B, \Delta} \quad \dfrac{\dfrac{\mathcal{D}_3}{\Gamma, B \vdash_{\text{LK}} \Delta}}{\Gamma, A, B \vdash_{\text{LK}} \Delta}\,(W)}{\Gamma, A \vdash_{\text{LK}} \Delta}\,(cut)}{\Gamma \vdash_{\text{LK}} \Delta}\,(cut)$$

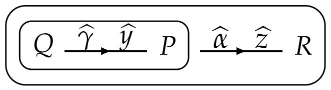
the witness for the first proof, $(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R)$



reduces to both $Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R)$ and $(Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R$, being the witnesses for the two resulting proofs:



and



This behaviour is reflected in rule $(exp\text{-}imp)$, as presented in Definition 2.2. We can see the *cut* $(\widehat{y}P\widehat{\alpha}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R)$ as a function $\widehat{y}P\widehat{\alpha}{\cdot}\gamma$ (with body P, that takes input on x and outputs on

α) interacting with a context $Q\widehat{\gamma}\,[x]\,\widehat{z}R$ (consisting of the function's argument Q, x as the hole that the function should occupy, and the context of this function application R)[4]. The contraction of the *cut* expresses (in the left-hand diagram) that the body of the function (which represents the result of the function, but with the substitution of the argument still pending) interacts with the context before using the argument; the other contraction first uses the argument, before interacting with the context, which corresponds to the standard way[5].

The calculus, defined by the reduction rules below, explains in detail how *cuts* are propagated through terms to be eventually evaluated at the level of *capsules,* where renaming takes place. Reduction is defined by specifying both the interaction between well-connected basic syntactic structures, and how to deal with propagating active nodes to points in the term where they can interact.

It is important to know when a connector is introduced, *i.e.* is connectable, *i.e.* is exposed and unique; this will play an important role in the reduction rules. Informally, a term P introduces a socket x if P is constructed from sub-terms which do not contain x as free socket, so x only occurs at the "top level." This means that P is either an *import* with a middle connector $[x]$ or a *capsule* with left part x. Similarly, a term introduces a plug α if it is an *export* that "creates" α or a *capsule* with right part α.

Definition 2.1 (Introduction)**.** P **introduces** x**:** Either $P = Q\widehat{\beta}\,[x]\,\widehat{y}R$ with $x \notin fs(Q,R)$, or $P = \langle x{\cdot}\alpha\rangle$.

P **introduces** α**:** Either $P = \widehat{x}Q\widehat{\beta}{\cdot}\alpha$ and $\alpha \notin fp(Q)$, or $P = \langle x{\cdot}\alpha\rangle$.

The principal reduction rules specify how to reduce a term that *cuts* sub-terms which introduce connectors. These rules are naturally divided in four categories: when a *capsule* is *cut* with a *capsule*, an *export* with a *capsule*, a *capsule* with an *import* or an *export* with an *import*. There is no other pattern in which a plug is introduced on the left of a $\dagger$ and a socket is introduced on the right.

Definition 2.2 (Logical rules)**.** Let α and x be introduced in, respectively, the left and right-hand side of the main *cuts* below.

$$\begin{array}{rrcl}
(\mathit{cap}): & \langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\beta\rangle & \to & \langle y{\cdot}\beta\rangle \\
(\mathit{exp}): & (\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle & \to & \widehat{y}P\widehat{\beta}{\cdot}\gamma \\
(\mathit{imp}): & \langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{z}R) & \to & Q\widehat{\beta}\,[y]\,\widehat{z}R \\
(\mathit{exp\text{-}imp}): & (\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) & \to & \begin{cases} Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R) \\ (Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R \end{cases}
\end{array}$$

The first three logical rules above specify a renaming procedure, whereas the last rule specifies the basic computational step: it links the *export* of a function, available on the plug α, to an adjacent *import* via the socket x. The effect of the reduction will be that the exported function is placed in-between the two sub-terms of the *import*, acting as interface. Notice that two *cuts* are created in the result, that can be grouped in two ways; these alternatives do not necessarily share all normal forms (reduction is non-confluent, so normal forms are not unique).

We now define how to reduce a *cut* when one of its sub-terms does *not* introduce a connector mentioned in the *cut*. This will involve moving the *cut* inwards, towards a position where the connector *is* introduced, with the direction taken indicated by the tilting of the dagger. In case both

[4]This view is confirmed by $\overline{\lambda}\mu\tilde{\mu}$, where $[\![P\widehat{\alpha} \dagger \widehat{x}Q]\!]^{\mathcal{X}} = \langle \mu\alpha.[\![P]\!]^{\mathcal{X}} \mid \tilde{\mu}x.[\![Q]\!]^{\mathcal{X}}\rangle$; in a *command* $\langle v \mid e\rangle$, v is a *term,* and e is a *context.*

[5]In fact, in $\overline{\lambda}\mu\tilde{\mu}$ only the second alternative is represented.

connectors are not introduced, this search can start in either direction, giving another source of non-confluence.

Definition 2.3 (Active *cuts*). The syntax is extended with two *flagged* or *active cuts*:

$$P ::= \ldots \mid P_1\widehat{\alpha} \nearrow\!\!\!\!\dagger\, \widehat{x}P_2 \mid P_1\widehat{\alpha} \nwarrow\!\!\!\!\dagger\, \widehat{x}P_2$$

Terms constructed without these flagged *cuts* are called *pure*.

We define two *cut*-activation rules.

$$\begin{array}{lrcll}
(a\nearrow\!\!\!\!\dagger): & P\widehat{\alpha} \dagger \widehat{x}Q & \to & P\widehat{\alpha} \nearrow\!\!\!\!\dagger\, \widehat{x}Q & \textit{if } P \textit{ does not introduce } \alpha \\
(\nwarrow\!\!\!\!\dagger a): & P\widehat{\alpha} \dagger \widehat{x}Q & \to & P\widehat{\alpha} \nwarrow\!\!\!\!\dagger\, \widehat{x}Q & \textit{if } Q \textit{ does not introduce } x
\end{array}$$

Notice that both side-conditions can hold simultaneously.

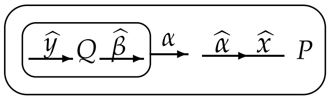
Similarly to the reasoning above, also the rules dealing with activated *cuts* are inspired by Gentzen's *cut*-elimination rules. Since

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_1\\ \Gamma, A \vdash_{\mathrm{LK}} A{\to}B, B, \Delta\end{array}}{\Gamma \vdash_{\mathrm{LK}} A{\to}B, \Delta}\,(\to R) \qquad \begin{array}{c}\mathcal{D}_2\\ \Gamma, A{\to}B \vdash_{\mathrm{LK}} \Delta\end{array}}{\Gamma \vdash_{\mathrm{LK}} \Delta}\,(cut)$$

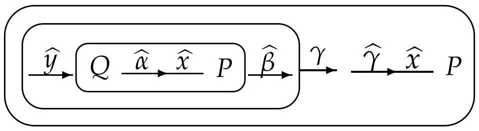
(notice the contraction towards $A{\to}B$ in the left-hand sub-derivation, so the plug associated to this formula would not be introduced in the witness for $\Gamma \vdash_{\mathrm{LK}} A{\to}B, \Delta$) contracts to

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_1\\ \Gamma, A \vdash_{\mathrm{LK}} A{\to}B, B, \Delta\end{array} \qquad \begin{array}{c}\mathcal{D}_2\\ \Gamma, A{\to}B \vdash_{\mathrm{LK}} \Delta\end{array}}{\Gamma, A \vdash_{\mathrm{LK}} B, \Delta}\,(cut)}{\Gamma \vdash_{\mathrm{LK}} A{\to}B, \Delta}\,(\to R) \qquad \begin{array}{c}\mathcal{D}_2\\ \Gamma, A{\to}B \vdash_{\mathrm{LK}} \Delta\end{array}}{\Gamma \vdash_{\mathrm{LK}} \Delta}\,(cut)$$

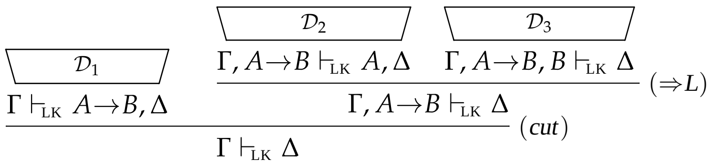
Notice that now in the conclusion of the left-hand sub-derivation the formula $A{\to}B$ is not contracted: in the witness for this proof, this corresponds to an introduced plug; in fact, the witness for the first proof, the term $(\widehat{y}Q\widehat{\beta}\cdot\alpha)\widehat{\alpha} \nearrow\!\!\!\!\dagger\, \widehat{x}P$, reduces to the witness for the second proof $(\widehat{y}(Q\widehat{\alpha} \nearrow\!\!\!\!\dagger\, \widehat{x}P)\widehat{\beta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{x}P$ where now γ is introduced, as reflected in rule $(\textit{exp-outs}\nearrow\!\!\!\!\dagger)$ below. So the diagram



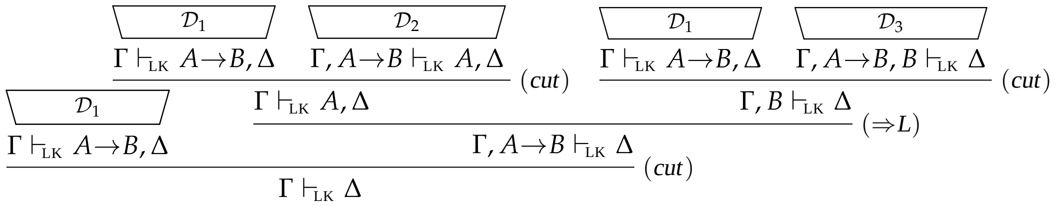
with α free in Q, reduces to



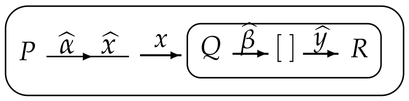
Also, since



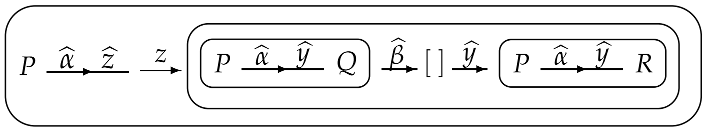
(again, notice the contraction) reduces to



the term $P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R)$ reduces to $P\widehat{\alpha}\dagger\widehat{z}((P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}R))$, or:



(where x occurs free in Q or R) reduces to



as reflected in rule $(\nwarrow\!\!\!\dagger\,\textit{imp-outs})$.

The next rules define how to move an activated dagger inwards.

Definition 2.4 (Propagation rules)**.** Left propagation:

$$\begin{array}{rrcll}
(d\,\dagger\!\!\!\nearrow): & \langle y{\cdot}\alpha\rangle\widehat{\alpha}\dagger\!\!\!\nearrow\widehat{x}P & \to & \langle y{\cdot}\alpha\rangle\widehat{\alpha}\dagger\widehat{x}P & \\
(\textit{cap}\,\dagger\!\!\!\nearrow): & \langle y{\cdot}\beta\rangle\widehat{\alpha}\dagger\!\!\!\nearrow\widehat{x}P & \to & \langle y{\cdot}\beta\rangle & \beta\neq\alpha \\
(\textit{exp-outs}\,\dagger\!\!\!\nearrow): & (\widehat{y}Q\widehat{\beta}{\cdot}\alpha)\widehat{\alpha}\dagger\!\!\!\nearrow\widehat{x}P & \to & (\widehat{y}(Q\widehat{\alpha}\dagger\!\!\!\nearrow\widehat{x}P)\widehat{\beta}{\cdot}\gamma)\widehat{\gamma}\dagger\widehat{x}P & \gamma\ \textit{fresh} \\
(\textit{exp-ins}\,\dagger\!\!\!\nearrow): & (\widehat{y}Q\widehat{\beta}{\cdot}\gamma)\widehat{\alpha}\dagger\!\!\!\nearrow\widehat{x}P & \to & \widehat{y}(Q\widehat{\alpha}\dagger\!\!\!\nearrow\widehat{x}P)\widehat{\beta}{\cdot}\gamma & \gamma\neq\alpha \\
(\textit{imp}\,\dagger\!\!\!\nearrow): & (Q\widehat{\beta}\,[z]\,\widehat{y}R)\widehat{\alpha}\dagger\!\!\!\nearrow\widehat{x}P & \to & (Q\widehat{\alpha}\dagger\!\!\!\nearrow\widehat{x}P)\widehat{\beta}\,[z]\,\widehat{y}(R\widehat{\alpha}\dagger\!\!\!\nearrow\widehat{x}P) & \\
(\textit{cut}\,\dagger\!\!\!\nearrow): & (Q\widehat{\beta}\dagger\widehat{y}R)\widehat{\alpha}\dagger\!\!\!\nearrow\widehat{x}P & \to & (Q\widehat{\alpha}\dagger\!\!\!\nearrow\widehat{x}P)\widehat{\beta}\dagger\widehat{y}(R\widehat{\alpha}\dagger\!\!\!\nearrow\widehat{x}P) &
\end{array}$$

Right propagation:

$$\begin{array}{lllll}
(\nwarrow\!\!\!\dagger\,d): & P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}\langle x{\cdot}\beta\rangle & \to & P\widehat{\alpha}\dagger\widehat{x}\langle x{\cdot}\beta\rangle & \\
(\nwarrow\!\!\!\dagger\,\textit{cap}): & P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}\langle y{\cdot}\beta\rangle & \to & \langle y{\cdot}\beta\rangle & y\neq x \\
(\nwarrow\!\!\!\dagger\,\textit{exp}): & P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}(\widehat{y}Q\widehat{\beta}{\cdot}\gamma) & \to & \widehat{y}(P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}Q)\widehat{\beta}{\cdot}\gamma & \\
(\nwarrow\!\!\!\dagger\,\textit{imp-outs}): & P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R) & \to & P\widehat{\alpha}\dagger\widehat{z}((P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}R)), & z\ \textit{fresh} \\
(\nwarrow\!\!\!\dagger\,\textit{imp-ins}): & P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}(Q\widehat{\beta}\,[z]\,\widehat{y}R) & \to & (P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}R) & z\neq x \\
(\nwarrow\!\!\!\dagger\,\textit{cut}): & P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}(Q\widehat{\beta}\dagger\widehat{y}R) & \to & (P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}Q)\widehat{\beta}\dagger\widehat{y}(P\widehat{\alpha}\nwarrow\!\!\!\dagger\,\widehat{x}R) &
\end{array}$$

Notice that, in rules $(d\,\dagger\!\!\!\nearrow)$ and $(\nwarrow\!\!\!\dagger\,d)$, the activated *cut* gets deactivated: although the connector mentioned in the *capsule* is certainly introduced, we cannot guarantee that the other connector is not, so it might be possible that now a logical rule is applicable; if the other connector is not introduced, the *cut* gets activated again, but now in the opposite direction.

Definition 2.5. (1) We write $\rightarrow_{\mathcal{X}}$ for the reduction relation defined as the smallest pre-order (*i.e.* reflexive and transitive relation) that includes the logical, propagation and activation rules, extended with the contextual rules[6]

$$P \rightarrow Q \;\Rightarrow\; \begin{cases} \widehat{x}P\widehat{\alpha}\cdot\beta & \rightarrow \;\widehat{x}Q\widehat{\alpha}\cdot\beta \\ P\widehat{\alpha}\,[x]\,\widehat{y}R & \rightarrow \;Q\widehat{\alpha}\,[x]\,\widehat{y}R \\ R\widehat{\alpha}\,[x]\,\widehat{y}P & \rightarrow \;R\widehat{\alpha}\,[x]\,\widehat{y}Q \\ P\widehat{\alpha}\dagger\widehat{y}R & \rightarrow \;Q\widehat{\alpha}\dagger\widehat{y}R \\ R\widehat{\alpha}\dagger\widehat{y}P & \rightarrow \;R\widehat{\alpha}\dagger\widehat{y}Q \end{cases}$$

The reduction $\rightarrow_{\mathcal{X}}$ is not confluent; this comes in fact from the critical pair that activates a *cut* $P\widehat{\alpha}\dagger\widehat{x}Q$ in two ways. Confluent sub-reduction systems are defined in [10].

Summarising, reduction brings all *cuts* down to logical *cuts* where both connectors are single and introduced, or to the elimination of *cuts* that are cutting towards a *capsule* that does not contain the relevant connector. Cuts towards connectors occurring in *capsules* lead to renaming $P\widehat{\alpha} \nwarrow \widehat{x}\langle x{\cdot}\beta\rangle \rightarrow_{\mathcal{X}} P[\beta/\alpha]$ and $\langle z{\cdot}\alpha\rangle\widehat{\alpha} \nearrow \widehat{x}P \rightarrow_{\mathcal{X}} P[z/x]$, and towards non-occurring connectors leads to elimination ($P\widehat{\alpha} \nwarrow \widehat{x}\langle z{\cdot}\beta\rangle \rightarrow_{\mathcal{X}} \langle z{\cdot}\beta\rangle$ and $\langle z{\cdot}\beta\rangle\widehat{\alpha} \nearrow \widehat{x}P \rightarrow_{\mathcal{X}} \langle z{\cdot}\beta\rangle$).

We remark that it is possible to define *cut*-elimination in many ways, and that the above rules are not cast in iron, but form a very elegant, natural and minimal set. We could, for example, replace the deactivation rules $(d\nearrow)$ and $(\nwarrow d)$ by $\langle z{\cdot}\alpha\rangle\widehat{\alpha} \nearrow \widehat{x}P \rightarrow P[z/x]$ and $P\widehat{\alpha} \nwarrow \widehat{x}\langle x{\cdot}\beta\rangle \rightarrow P[\beta/\alpha]$, respectively; this yields $\mathcal{X}^i$, a variant of $\mathcal{X}$ with implicit substitution as defined in [43]. The activated *cuts* were introduced by Urban with the main purpose of giving enough control over *cut*-elimination to prove strong normalisation, without sacrificing expressivity. The idea is that, once activated, a *cut* has to run to completion, and cannot be "crossed" with another *cut*.

The soundness result of simple type assignment with respect to reduction is stated as usual:

Theorem 2.6 (Witness reduction [10])**.** *If* $P :\cdot \Gamma \vdash \Delta$, *and* $P \rightarrow_{\mathcal{X}} Q$, *then* $Q :\cdot \Gamma \vdash \Delta$.

In [10, 11] some basic properties are shown, which essentially show that the calculus is well behaved, as well as the relation between $\mathcal{X}$ and a number of other calculi. These results motivate the formulation of admissible rules:

Lemma 2.7 (Garbage Collection and Renaming [11])**.**

$$\begin{array}{llll} (\nearrow gc): & P\widehat{\alpha} \nearrow \widehat{x}Q \rightarrow_{\mathcal{X}} P & \text{if } \alpha \notin fp(P) & \quad (ren\text{-L}): \;\langle z{\cdot}\alpha\rangle\widehat{\alpha}\dagger\widehat{x}P \rightarrow_{\mathcal{X}} P[z/x] \\ (\nwarrow gc): & P\widehat{\alpha} \nwarrow \widehat{x}Q \rightarrow_{\mathcal{X}} Q & \text{if } x \notin fs(Q) & \quad (ren\text{-R}): \;P\widehat{\delta}\dagger\widehat{z}\langle z{\cdot}\alpha\rangle \rightarrow_{\mathcal{X}} P[\alpha/\delta] \end{array}$$

Example 2.8. To illustrate reduction in $\mathcal{X}$, we will reduce the term

$$(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma}\dagger\widehat{u}(\langle u{\cdot}\beta\rangle\widehat{\beta}\dagger\widehat{y}(Q\widehat{\tau}\,[y]\,\widehat{w}R))$$

where $P = \langle z{\cdot}\delta\rangle$, $Q = \langle v{\cdot}\tau\rangle$ and $R = \langle w{\cdot}\sigma\rangle$, so $\gamma \notin fp(P)$ and $u, y \notin fs(Q, R)$. Notice that, since u not introduced in the right-hand term, this is not a logical *cut*. We show two reduction paths; in

[6]Since reduction in $\mathcal{X}$ is defined via rewriting rules, the contextual rules are normally left implicit; we mention them here because we will define a restriction of reduction that also limits the contextual rules.

the first reduction, we contract first the innermost (logical) *cut*:

$$\begin{array}{lll}
(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{u}(\langle u\cdot\beta\rangle\widehat{\beta} \dagger \widehat{y}(Q\widehat{\tau}\,[y]\,\widehat{w}R)) & \to & (\mathit{imp}) \\
(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{u}(Q\widehat{\tau}\,[u]\,\widehat{w}R) & \to & (\mathit{exp\text{-}imp}) \\
Q\widehat{\tau} \dagger \widehat{z}(P\widehat{\delta} \dagger \widehat{w}R) & &
\end{array}$$

We could run this further, but for our purposes this is enough.

In the second, we first activate the outer-most *cut*:

$$\begin{array}{lll}
(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{u}(\langle u\cdot\beta\rangle\widehat{\beta} \dagger \widehat{y}(Q\widehat{\tau}\,[y]\,\widehat{w}R)) & \to & (\nwarrow a, \nwarrow \mathit{cut}) \\
((\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \nwarrow \widehat{u}\langle u\cdot\beta\rangle)\widehat{\beta} \dagger \widehat{y}((\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \nwarrow \widehat{u}(Q\widehat{\tau}\,[y]\,\widehat{w}R)) & \to & (\nwarrow d, \mathit{exp}, \nwarrow \mathit{imp\text{-}ins}) \\
(\widehat{z}P\widehat{\delta}\cdot\beta)\widehat{\beta} \dagger \widehat{y}(((\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \nwarrow \widehat{u}Q)\widehat{\tau}\,[y]\,\widehat{w}((\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \nwarrow \widehat{u}R)) & \to & (\nwarrow \mathit{cap}) \\
(\widehat{z}P\widehat{\delta}\cdot\beta)\widehat{\beta} \dagger \widehat{y}(Q\widehat{\tau}\,[y]\,\widehat{w}R) & \to & (\mathit{exp\text{-}imp}) \\
Q\widehat{\tau} \dagger \widehat{z}(P\widehat{\delta} \dagger \widehat{w}R) & &
\end{array}$$

For another example, let $P = \langle z\cdot\delta\rangle$, $Q' = \langle u\cdot\tau\rangle$ and $R = \langle w\cdot\sigma\rangle$ (notice the difference in Q, so u is no longer introduced in $Q'\widehat{\tau}\,[u]\,\widehat{w}R$).

$$\begin{array}{lll}
(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{u}(Q'\widehat{\tau}\,[u]\,\widehat{w}R) & \to & (\nwarrow a) \\
(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \nwarrow \widehat{u}(Q'\widehat{\tau}\,[u]\,\widehat{w}R) & \to & (\nwarrow \mathit{imp\text{-}outs}) \\
(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{y}(((\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \nwarrow \widehat{u}Q')\widehat{\tau}\,[y]\,\widehat{w}((\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \nwarrow \widehat{u}R)) & \to & (\nwarrow d, \nwarrow \mathit{cap}) \\
(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{y}(((\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{u}Q')\widehat{\tau}\,[y]\,\widehat{w}R) & \to & (\mathit{exp}), =_\alpha \\
(\widehat{x}\langle x\cdot\rho\rangle\widehat{\rho}\cdot\gamma)\widehat{\gamma} \dagger \widehat{y}((\widehat{z}P\widehat{\delta}\cdot\tau)\widehat{\tau}\,[y]\,\widehat{w}R) & \to & (\mathit{exp\text{-}imp}) \\
(\widehat{z}P\widehat{\delta}\cdot\tau)\widehat{\tau} \dagger \widehat{x}(\langle x\cdot\rho\rangle\widehat{\rho} \dagger \widehat{w}R) & \to & (\nwarrow a) \\
(\widehat{z}P\widehat{\delta}\cdot\tau)\widehat{\tau} \nwarrow \widehat{x}(\langle x\cdot\rho\rangle\widehat{\rho} \dagger \widehat{w}R) & \to & (\nwarrow \mathit{cut}) \\
((\widehat{z}P\widehat{\delta}\cdot\tau)\widehat{\tau} \nwarrow \widehat{z}\langle z\cdot r\rangle)\widehat{\rho} \dagger \widehat{w}((\widehat{z}P\widehat{\delta}\cdot\tau)\widehat{\tau} \nwarrow \widehat{z}R) & \to & (\nwarrow d, \mathit{exp}, \nwarrow \mathit{cap}) \\
(\widehat{z}P\widehat{\delta}\cdot\rho)\widehat{\rho} \dagger \widehat{w}R & \to & (\mathit{exp}) \\
\widehat{z}P\widehat{\delta}\cdot\sigma & &
\end{array}$$

Example 2.9. We show how to reduce a *cut* containing the witness for Peirce's law in a context that offers identity as a first argument:

$$\begin{array}{lll}
(\widehat{z}((\widehat{y}\langle y\cdot\delta\rangle\widehat{\eta}\cdot\alpha)\widehat{\alpha}\,[z]\,\widehat{v}\langle v\cdot\delta\rangle)\widehat{\delta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{v}((\widehat{x}\langle x\cdot\sigma\rangle\widehat{\sigma}\cdot\tau)\widehat{\tau}\,[v]\,\widehat{w}\langle w\cdot\rho\rangle) & \to & (\mathit{exp\text{-}imp}) \\
(\widehat{x}\langle x\cdot\sigma\rangle\widehat{\sigma}\cdot\tau)\widehat{\tau} \dagger \widehat{z}(((\widehat{y}\langle y\cdot\delta\rangle\widehat{\eta}\cdot\alpha)\widehat{\alpha}\,[z]\,\widehat{v}\langle v\cdot\delta\rangle)\widehat{\delta} \dagger \widehat{w}\langle w\cdot\rho\rangle) & \to & (\mathit{ren}\text{-}\mathrm{R}) \\
(\widehat{x}\langle x\cdot\sigma\rangle\widehat{\sigma}\cdot\tau)\widehat{\tau} \dagger \widehat{z}((\widehat{y}\langle y\cdot\rho\rangle\widehat{\eta}\cdot\alpha)\widehat{\alpha}\,[z]\,\widehat{v}\langle v\cdot\rho\rangle) & \to & (\mathit{exp\text{-}imp}) \\
(\widehat{y}\langle y\cdot\rho\rangle\widehat{\eta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}(\langle x\cdot\sigma\rangle\widehat{\sigma} \dagger \widehat{v}\langle v\cdot\rho\rangle) & \to & (\mathit{cap}) \\
(\widehat{y}\langle y\cdot\rho\rangle\widehat{\eta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x\cdot\rho\rangle & \to & (\mathit{exp}) \\
\widehat{y}\langle y\cdot\rho\rangle\widehat{\eta}\cdot\rho & &
\end{array}$$

Notice that we cannot type these terms: as shown in Example 1.5, the type used for z in the subterm $(\widehat{y}\langle y\cdot\delta\rangle\widehat{\eta}\cdot\alpha)\widehat{\alpha}\,[z]\,\widehat{v}\langle v\cdot\delta\rangle$ is $(A\to B)\to A$, and we cannot assign that type to τ in $\widehat{x}\langle x\cdot\sigma\rangle\widehat{\sigma}\cdot\tau$, since that term is only a witness of judgements of the shape $\widehat{x}\langle x\cdot\sigma\rangle\widehat{\sigma}\cdot\tau \;\vdots\cdot\; \Gamma \vdash_{\mathcal{X}} \tau{:}C\to C$, and we cannot solve $(A\to B)\to A = C\to C$. And in fact, to type the final term $\widehat{y}\langle y\cdot\rho\rangle\widehat{\eta}\cdot\rho$, the connector ρ must have *both* the types C and $C\to D$, which we cannot express.

3. THE ASYNCHRONOUS π-CALCULUS WITH PAIRING

The notion of asynchronous π-calculus that we consider in this paper is different from other systems studied in the literature [31], and can be seen as a special case of the polyadic π-calculus [37]; the reason for this deviation is made clear in Section 4. One reason for this change lies directly in the calculus that is going to be interpreted, $\mathcal{X}$, in which a term can be constructed binding *two names simultaneously*; we will model this via the sending and receiving pairs of names as interfaces for functions, so, inspired by [2], add pairing. We take the view that processes communicate by sending data over channels, so not just names, but also pairs of names.

We will define an encoding of $\mathcal{X}$ into this π-calculus with pairing. Almost as usual, we cannot model full *cut*-elimination through our first encoding of $\mathcal{X}$-terms via $[\![\cdot]\!]_{s}$ into the π-calculus; this is directly caused by the nature of the reduction relation on the π-calculus, which does not permit reduction under an *input*. This was also the case with the interpretations of the λ-calculus defined by, for example, Milner [36], Sangiorgi [41], Honda *et al.* [33], Thielecke [44], and two of the authors of this paper [13], where reduction in the original calculus had to be restricted in order to get a completeness result. However, we will be able to overcome that shortcoming, and define two encodings that *do* represent $\mathcal{X}$'s reduction in full.

To ease the definition of the interpretation function of terms in $\mathcal{X}$ to processes in the π-calculus, we deviate slightly from the normal practice, and write either Greek characters $\alpha, \beta, v, \ldots$ or Roman characters $x, y, z, \ldots$ for channel names; we use n for either a Greek or a Roman name. To successfully preserve assignable types, 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.

Definition 3.1 (Asynchronous π-calculus)**.** (1) Channel names and data are defined by:

$$\begin{array}{rcll} a,b,c,d & ::= & x \mid \alpha & \textit{names} \\ p & ::= & a \mid \langle a,b \rangle & \textit{data} \end{array}$$

Notice that pairing is *not* recursive.

(2) Processes are defined by:

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

(3) We abbreviate $a(x).\,\textit{let}\,\langle y,z \rangle = x \textit{ in } P$ by $a(y,z).\,P$, and $(\nu m)\,(\nu n)\,P$ by $(\nu mn)\,P$, and write $\overline{a}\langle c,d \rangle$ rather than $\overline{a}\langle\langle c,d \rangle\rangle$.

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

(5) We consider n bound in $(\nu n)\,P$, and call n free in P if it occurs in P and is not bound; we write $\textit{fn}(P)$ for the set of free names in P, and write $\textit{fn}(P,Q)$ for $\textit{fn}(P) \cup \textit{fn}(P)$.

Definition 3.2 (Congruence)**.** The *structural congruence* is the smallest equivalence relation closed under contexts defined by the following rules:

$$P \mid \mathbf{0} \equiv P \qquad !\,P \equiv P \mid !\,P \qquad !\,P \equiv !\,P \mid !\,P \qquad P \mid Q \equiv Q \mid P \qquad (\nu n)\,\mathbf{0} \equiv \mathbf{0}$$

$$\begin{array}{rclcrcl} (\nu m)\,(\nu n)\,P & \equiv & (\nu n)\,(\nu m)\,P & & (\nu n)\,(P \mid Q) & \equiv & P \mid (\nu n)\,Q \quad \textit{if } n \notin \textit{fn}(P) \\ (P \mid Q) \mid R & \equiv & P \mid (Q \mid R) & & \textit{let}\,\langle x,y \rangle = \langle a,b \rangle \textit{ in } R & \equiv & R[a/x, b/y] \end{array}$$

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.

Definition 3.3 (Reduction)**.** (1) The *reduction relation* over the processes of the π-calculus is defined by following (elementary) rules:

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

(2) We write $\to_\pi^+$ for the transitive closure of $\to_\pi$, and $\to_\pi^*$ for the reflexive and transitive closure of $\to_\pi$.

Notice that
$$\begin{array}{rcl}
\overline{a}\langle b,c\rangle \,|\, a(x,y).\,Q & = & \overline{a}\langle b,c\rangle \,|\, a(z).\,\mathit{let}\,\langle x,y\rangle = z \;\mathit{in}\; Q\;. \\
 & \to_\pi & \mathit{let}\,\langle x,y\rangle = \langle b,c\rangle \;\mathit{in}\; Q \\
 & \equiv & Q[b/x, c/y]
\end{array}$$

Definition 3.4. (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 $P \to_\pi^* Q$ 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 $P \sim_{\mathsf{c}} Q$ (and call P and Q *contextually equivalent*) if and only if $P \sqsubseteq_{\mathsf{c}} Q$ and $Q \sqsubseteq_{\mathsf{c}} P$.

Definition 3.5. (1) *Strong equivalence* is the largest relation $\stackrel{\bullet}{\sim}$ such that $P \stackrel{\bullet}{\sim} Q$ implies:

- for each name n, $P \downarrow n$ if and only if $Q \downarrow n$;
- for all P', if $P \to_\pi P'$, then for some Q', $Q \to_\pi Q'$ and $P' \sim_\pi Q'$.
- for all Q', if $\mathsf{C}[Q] \to_\pi Q'$, then for some P', $P \to_\pi P'$ and $Q' \stackrel{\bullet}{\sim} P'$.

(2) *Strong bisimilarity* is the largest relation $\sim$ such that $P \sim Q$ if for all processes R, $P \,|\, R \stackrel{\bullet}{\sim} Q \,|\, R$.

Theorem 3.6 ([40])**.** *(1) $\sim$ is a congruence relation.*

(2) $\sim$ implies $\sim_{\mathsf{c}}$.

The following lemma was shown in [41] using $\sim$, and states some basic properties on processes that are relevant to our results; especially the second and third, that state distribution rules, are important.

Lemma 3.7 (*cf.* [41])**.** *(1) (a)* $(\nu x)\,(!\,P) \sim_{\mathsf{c}} \,!\,(\nu x)\,(!\,P)$.

(b) $(\nu x)\,(!\,Q \,|\, !\,P) \;\sim_{\mathsf{c}}\; !\,(\nu x)\,(Q \,|\, !\,P)$

(2) Let Q, R be processes that use α only for output, and P has α only as input. Then:

(a) $(\nu\alpha)\,(Q \,|\, R \,|\, !\,P) \;\sim_{\mathsf{c}}\; (\nu\alpha)\,(Q \,|\, !\,P) \,|\, (\nu\alpha)\,(R \,|\, !\,P)$

(b) $(\nu\alpha)\,((\nu\beta)\,(Q \,|\, R) \,|\, !\,P) \;\sim_{\mathsf{c}}\; (\nu\gamma)\,((\nu\beta)\,((\nu\alpha)\,(Q \,|\, !\,P)) \,|\, R[\gamma/\alpha] \,|\, !\,P[\gamma/\alpha])$

(3) Let Q, R be processes that use x only for input, and P has x only as output. Then:

(a) $(\nu x)\,(Q \,|\, R \,|\, !\,P) \;\sim_{\mathsf{c}}\; (\nu x)\,(Q \,|\, !\,P) \,|\, (\nu x)\,(R \,|\, !\,P)$

(b) $(\nu x)\,((\nu y)\,(Q \,|\, R) \,|\, !\,P) \;\sim_{\mathsf{c}}\; (\nu z)\,((\nu y)\,((\nu x)\,(Q \,|\, !\,P)) \,|\, R[z/x] \,|\, !\,P[z/x])$

4. A SIMPLE ENCODING FOR $\mathcal{X}$ INTO π

In this section we will present an encoding of $\mathcal{X}$ into π which closely follows the structure and intuition of $\mathcal{X}$. Our encoding is based on the intuition formulated above: the *cut* $P\widehat{\alpha} \dagger \widehat{x}Q$ expresses the intention to connect all αs in P and xs in Q. Translated into π, this results in seeing P as trying to send at least as many times over α as Q is willing to receive over x, and Q trying to receive at least as many times over x as P is ready to send over α.

Since some sub-terms will be placed under *input*, a full representation of reduction in $\mathcal{X}$ cannot be achieved, because it is not possible to reduce the (interpreted) terms that appear under an *input*; prohibiting reduction under *input* is necessary for the expansion lemma [41]. In view of the literature that exists on encodings into the π-calculus, this is unfortunate but normal: this limitation was already evident in [36], which manages only to show a preservation result for *lazy* reduction [3] for the λ-calculus, and is also present in [13] where only the notion of spine reduction gets represented. To accommodate for this shortcoming, to achieve a simulation result using this first encoding, we restrict the notion of reduction on $\mathcal{X}$ to that of *head*-reduction. As can be seen in Definition 4.1, *input* is only used for the encoding of *import*, so the restriction will consist of removing the rules that reduce under *import*; also, since no congruence rules exist that deal with *input* terms, propagation into an *import* cannot be modelled.

Although departing from $\mathcal{X}$ it is natural to use Greek names for outputs and Roman names for inputs, by the very nature of the communication of the π-calculus (it is only possible to communicate using the *same* channel for in and output), in the implementation we are forced to use Greek names also for inputs, and Roman names for outputs; in fact, we need to explicitly convert '*an output sent on α is to be received as input on x*' via '$\alpha(w).\overline{x}\langle w\rangle$' (called a *forwarder* in [32]), so α is now also an input, and x also an output channel, which for convenience is abbreviated into $\alpha{\rightarrow}x$.

Definition 4.1 (Simple interpretation of $\mathcal{X}$ in π)**.** The *simple* interpretation of terms is defined by:

$$\begin{array}{rcll}
[\![\langle x{\cdot}\alpha\rangle]\!]_{\mathrm{s}} &=& x(w).\overline{\alpha}\langle w\rangle & \\
[\![\widehat{y}Q\widehat{\beta}{\cdot}\alpha]\!]_{\mathrm{s}} &=& (\nu y\beta)\,(!\,[\![Q]\!]_{\mathrm{s}} \mid \overline{\alpha}\langle y,\beta\rangle) & \\
[\![P\widehat{\alpha}\,[x]\,\widehat{y}Q]\!]_{\mathrm{s}} &=& x(s,d).\,((\nu\alpha)\,(!\,[\![P]\!]_{\mathrm{s}} \mid !\,\alpha{\rightarrow}s) \mid (\nu y)\,(!\,d{\rightarrow}y \mid !\,[\![Q]\!]_{\mathrm{s}})) & \\
[\![P\widehat{\alpha} \dagger \widehat{x}Q]\!]_{\mathrm{s}} &=& (\nu\alpha x)\,(!\,[\![P]\!]_{\mathrm{s}} \mid !\,\alpha{\rightarrow}x \mid !\,[\![Q]\!]_{\mathrm{s}}) & = [\![P\widehat{\alpha} \not\dagger \widehat{x}Q]\!]_{\mathrm{s}} = [\![P\widehat{\alpha} \setminus\!\!\!\dagger\, \widehat{x}Q]\!]_{\mathrm{s}}
\end{array}$$

The approach of $[\![\cdot]\!]_{\mathrm{s}}$ is to see the *import* $P\widehat{\alpha}\,[x]\,\widehat{y}Q$ as a delayed communication, that is waiting for a mediator to arrive in x. Notice that the term P outputs on α, and Q inputs on y, and that these are bound locally in the construction of the *import*, as correctly expressed by $(\nu\alpha)\,(!\,[\![P]\!]_{\mathrm{s}} \mid !\,\alpha{\rightarrow}s)$, where s is the name of an input channel that needs to be received, and $(\nu y)\,(!\,d{\rightarrow}y \mid !\,[\![Q]\!]_{\mathrm{s}})$, where d needs to be received; in fact, s and d are received together in a pair via *input*. Since we here follow quite closely the structure of terms in $\mathcal{X}$, it is this aspect that gives the moniker 'simple' to this encoding. Notice that we redirect the output α to s, which is the input channel of the mediator, and that $[\![P]\!]_{\mathrm{s}}$ gets replicated since it might be needed more than once in that mediator. We place these two processes in parallel under the input of $x(s,d)$, creating

$$x(s,d).\,((\nu\alpha)\,(!\,[\![P]\!]_{\mathrm{s}} \mid !\,\alpha{\rightarrow}s) \mid (\nu y)\,(!\,d{\rightarrow}y \mid !\,[\![Q]\!]_{\mathrm{s}})).$$

Since now sub-terms are placed under *input*, we cannot encode $\mathcal{X}$'s reduction in full.

As mentioned in the introduction, we added pairing to the π-calculus in order to be able to deal with arrow types. Notice that using the polyadic π-calculus instead would not be sufficient: since we would like the interpretation to respect reduction, in particular we need to be able to reduce the interpretation of $(\widehat{x}P\widehat{\alpha}{\cdot}\beta)\widehat{\beta} \dagger \widehat{z}\langle z{\cdot}\gamma\rangle$ to that of $\widehat{x}P\widehat{\alpha}{\cdot}\gamma$ (when β not free in P). So, choosing to encode the *export* of x and α over β as $\overline{\beta}\langle x,\alpha\rangle$ would force the interpretation of $\langle z{\cdot}\gamma\rangle$ to always receive a pair of names. But requiring for a *capsule* to always deal with pairs of names is too restrictive, since it is desirable to allow *capsules* to deal with single names as well. So, rather than moving towards the polyadic π-calculus, we opt for letting communication send a single item, which is either a name or a pair of names. This implies that a process sending a pair can also successfully communicate with a process not explicitly demanding to receive a pair.

Notice that the interpretation of non-activated *cuts* is the same as that of activated *cuts*; this implies that we are, in fact, also interpreting a variant of $\mathcal{X}$ *without* activated *cuts*, modelling arbitrary movement of *cuts* over *cuts*, but with the same set of rewrite rules. This is very different from Gentzen's original definition – he in fact does not define a *cut*-over-*cut* step, and uses innermost reduction for his *Hauptsatz* result – and different from Urban's definition: allowing only *activated cuts* to propagate is crucial for his Strong Normalisation result. However, this rewriting is still sound with respect to typeability, in the sense that assignable contexts are preserved under reduction. Here we can abstract from these aspects, since we only aim to prove a *simulation* result for $\mathcal{X}$, not full abstraction, for which the simple encoding will be shown adequate.

The following is straightforward:

Lemma 4.2 (Free name preservation). *$\alpha, x \notin fc\,(P)$, if and only if $\alpha, x \notin fn(\llbracket P \rrbracket_{\mathsf{s}})$.*

Proof. By easy induction on the structure of $\mathcal{X}$-terms. $\square$

There is a couple of important aspects to our result that need to be pointed out:

(1) One of the main goals we aimed for with our interpretation was, of course, the preservation of reduction: if $P \rightarrow_{\mathcal{X}} Q$, then $\llbracket P \rrbracket_{\mathsf{s}} \rightarrow_{\pi} \llbracket Q \rrbracket_{\mathsf{s}}$; we quickly understood that this was too ambitious. Take the reduction $\langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle \rightarrow_{\mathcal{X}} \langle y{\cdot}\gamma\rangle$, then

$$\begin{aligned}\llbracket \langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle \rrbracket_{\mathsf{s}} &\triangleq (\nu\alpha x)\,(!\,y(w).\overline{\alpha}\langle w\rangle \mid !\,\alpha{\twoheadrightarrow}x \mid !\,x(w).\overline{\gamma}\langle w\rangle)\\ \llbracket \langle y{\cdot}\gamma\rangle \rrbracket_{\mathsf{s}} &\triangleq y(w).\overline{\gamma}\langle w\rangle\end{aligned}$$

but we cannot show that $(\nu\alpha x)\,(!\,y(w).\overline{\alpha}\langle w\rangle \mid !\,\alpha{\twoheadrightarrow}x \mid !\,x(w).\overline{\gamma}\langle w\rangle)$ reduces to $y(w).\overline{\gamma}\langle w\rangle$. However, it is easy to show that these processes are *contextually equivalent*.

So, can we then show : if $P \rightarrow_{\mathcal{X}} Q$, then $\llbracket P \rrbracket_{\mathsf{s}} \sim_{\mathsf{c}} \llbracket Q \rrbracket_{\mathsf{s}}$? Alas, given the reduction rules in π, this cannot be achieved in full: since, in π, we cannot reduce under an input, we can only simulate head-reduction.

(2) Moreover, the reduction in $\mathcal{X}$ is non-confluent, so, in particular, a term P can have more than one normal form. When interpreting a term through its set of normal forms via $\ulcorner \cdot \lrcorner_{\mathrm{NF}}$, it is easy to show that, if $P \rightarrow_{\mathcal{X}} Q$, then $\ulcorner Q \lrcorner_{\mathrm{NF}} \subseteq \ulcorner P \lrcorner_{\mathrm{NF}}$; so picking one reduction from P can then exclude the reachability of some of the other normal forms, and the set of reachable normal forms decreases during reduction.

Something similar also holds for our encoding into the π-calculus: if $P \rightarrow_{\mathcal{X}} Q$, then $\llbracket P \rrbracket_{\mathsf{s}}$ has more behaviour than $\llbracket Q \rrbracket_{\mathsf{s}}$, expressed via $\llbracket P \rrbracket_{\mathsf{s}} \;{}_{\mathsf{c}}\!\sqsupseteq \llbracket Q \rrbracket_{\mathsf{s}}$.

We now define our notion of head-reduction on $\mathcal{X}$.

Definition 4.3. We define the notion of *head*-reduction $\rightarrow_{\mathrm{H}}$ as in Definition 2.5, by blocking reductions in and toward *import*, via the *elimination* of the propagation rules that move into an *import*:

$$\begin{array}{llcll}(\mathit{imp}\not\nearrow): & (Q\widehat{\beta}\,[z]\,\widehat{y}R)\widehat{\alpha} \not\nearrow \widehat{x}P & \rightarrow & (Q\widehat{\alpha} \not\nearrow \widehat{x}P)\widehat{\beta}\,[z]\,\widehat{y}(R\widehat{\alpha} \not\nearrow \widehat{x}P) & \\ (\not\nwarrow\mathit{imp\text{-}outs}): & P\widehat{\alpha} \not\nwarrow \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R) & \rightarrow & P\widehat{\alpha} \dagger \widehat{z}((P\widehat{\alpha} \not\nwarrow \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \not\nwarrow \widehat{x}R)) & \\ (\not\nwarrow\mathit{imp\text{-}ins}): & P\widehat{\alpha} \not\nwarrow \widehat{x}(Q\widehat{\beta}\,[z]\,\widehat{y}R) & \rightarrow & (P\widehat{\alpha} \not\nwarrow \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \not\nwarrow \widehat{x}R), & z \neq x\end{array}$$

as well as the contextual rules:

$$P \to Q \;\Rightarrow\; \begin{cases} P\widehat{\alpha}\,[x]\,\widehat{y}R \to Q\widehat{\alpha}\,[x]\,\widehat{y}R \\ R\widehat{\alpha}\,[x]\,\widehat{y}P \to R\widehat{\alpha}\,[x]\,\widehat{y}Q \end{cases}$$

The choice for the terminology *head*-reduction can be motivated as follows. The only remaining reduction rules that deal with *imports* are:

$$\begin{array}{lrcl} (\mathit{imp}): & \langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{z}R) & \to & Q\widehat{\beta}\,[y]\,\widehat{z}R \\ (\mathit{exp\text{-}imp}): & (\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) & \to & \begin{cases} Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R) \\ (Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R \end{cases} \end{array}$$

Take the logical *cut* $(\widehat{y}P\widehat{\alpha}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R)$; as mentioned above, this expresses a function $\widehat{y}P\widehat{\alpha}{\cdot}\gamma$ interacting with a context $Q\widehat{\gamma}\,[x]\,\widehat{z}R$, where Q is the function's parameter, and R is the context of this function application (we can compare this term, with discrepancies, to $(\lambda y.P)Q\overrightarrow{R_i}$, so R is the context $[\,]\overrightarrow{R_i}$). We can see the contraction of this *cut* as a substitution[7], where we insert P into the hole x in the context. The restriction we put on the rewriting system in head-reduction implies that this only will happen if the left-hand term mentioned in the *cut* is a value, *i.e.* either a *capsule* or an *export* $\widehat{y}P\widehat{\alpha}{\cdot}\gamma$ with α introduced. In particular, under head-reduction, in the term $T\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R)$ (which we can see as $xQ\overrightarrow{R_i}\,\langle x := T\rangle$, *i.e.* as $TQ\overrightarrow{R_i}$) all reduction takes place exclusively *inside* T (so in the head of the term), and the *cut* mentioned explicitly will only be contracted after that reduction produces a term that introduces α, either in a *capsule*, or in an *export*. So this substitution is postponed (for an introduced x; if x is not introduced, it will always be blocked, since propagation into an *import* is no longer allowed) until the term to be inserted has become a value; notice that reductions inside this value are allowed.

We now come to the correctness result for the encoding, which essentially states that we can mimic $\mathcal{X}$'s head-reduction in π: if $P \to_{\mathrm{H}} Q$, the image of the $\mathcal{X}$-term P under the encoding in π reduces to some π-process that contains the behaviour of Q, but might have some extra behaviour that could be disregarded. As is evident from the proofs below, this is in part due to the presence of replicated processes in the translation of the *cut*.

The precise formulation of the correctness result now becomes:

Theorem 4.4. *If $P \to_{\mathrm{H}} Q$, then $[\![P]\!]_{\mathrm{S}} \;{}_{\mathrm{C}}\!\sqsupseteq\; [\![Q]\!]_{\mathrm{S}}$.*

Proof. **Logical rules:**

(cap)**:** $\langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle \to \langle y{\cdot}\gamma\rangle$.

$$\begin{array}{lll} [\![\langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle]\!]_{\mathrm{S}} \triangleq & (\nu\alpha x)\,(!\,[\![\langle y{\cdot}\alpha\rangle]\!]_{\mathrm{S}} \mid !\,\alpha{\to}x \mid !\,[\![\langle x{\cdot}\gamma\rangle]\!]_{\mathrm{S}}) & \triangleq \\ & (\nu\alpha x)\,(!\,y(w).\overline{\alpha}\langle w\rangle \mid !\,\alpha{\to}x \mid !\,x(w).\overline{\gamma}\langle w\rangle) & {}_{\mathrm{C}}\!\sqsupseteq \\ & (\nu\alpha x)\,(y(w).\overline{\alpha}\langle w\rangle \mid \alpha{\to}x \mid x(w).\overline{\gamma}\langle w\rangle) & \sim_{\mathrm{C}}\;(\alpha,x) \\ & y(w).\overline{\gamma}\langle w\rangle & = \;[\![\langle y{\cdot}\gamma\rangle]\!]_{\mathrm{S}} \end{array}$$

(exp)**:** $(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle \to \widehat{y}P\widehat{\beta}{\cdot}\gamma$.

$$\begin{array}{ll} [\![(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle]\!]_{\mathrm{S}} \triangleq (\nu\alpha x)\,(!\,[\![\widehat{y}P\widehat{\beta}{\cdot}\alpha]\!]_{\mathrm{S}} \mid !\,\alpha{\to}x \mid !\,[\![\langle x{\cdot}\gamma\rangle]\!]_{\mathrm{S}}) & {}_{\mathrm{C}}\!\sqsupseteq \\ (\nu\alpha x)\,([\![\widehat{y}P\widehat{\beta}{\cdot}\alpha]\!]_{\mathrm{S}} \mid \alpha{\to}x \mid [\![\langle x{\cdot}\gamma\rangle]\!]_{\mathrm{S}}) & \triangleq \\ (\nu\alpha x)\,((\nu y\beta)\,(!\,[\![P]\!]_{\mathrm{S}} \mid \overline{\alpha}\langle y,\beta\rangle) \mid \alpha{\to}x \mid x(w).\overline{\gamma}\langle w\rangle) & \to_{\pi}^{+}\;(\alpha,x) \\ (\nu y\beta)\,(!\,[\![P]\!]_{\mathrm{S}} \mid \overline{\gamma}\langle y,\beta\rangle) & \triangleq\;[\![\widehat{y}P\widehat{\beta}{\cdot}\gamma]\!]_{\mathrm{S}} \end{array}$$

[7] In fact, it corresponds to a $\tilde{\mu}$-reduction in $\overline{\lambda}\mu\tilde{\mu}$, defined by the rule $\langle v \mid \tilde{\mu}x.c\rangle \to c[v/x]$, which performs the substitution.

(imp): $\langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{z}P) \to Q\widehat{\beta}\,[y]\,\widehat{z}P$.

$$
\begin{array}{ll}
[\![\langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{z}P)]\!]_{\mathsf{s}} \triangleq (\nu\alpha x)\,(!\,[\![\langle y{\cdot}\alpha\rangle]\!]_{\mathsf{s}} \mid !\,\alpha\to x \mid !\,[\![Q\widehat{\beta}\,[x]\,\widehat{z}P]\!]_{\mathsf{s}}) & {}_{\mathsf{c}}\!\sqsupseteq \\
(\nu\alpha x)\,([\![\langle y{\cdot}\alpha\rangle]\!]_{\mathsf{s}} \mid \alpha\to x \mid [\![Q\widehat{\beta}\,[x]\,\widehat{z}P]\!]_{\mathsf{s}}) & \triangleq \\
(\nu\alpha x)\,(y(w).\,\overline{\alpha}\langle w\rangle \mid \alpha\to x \mid x(s,d).\,((\nu\beta)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid !\,\beta\to s) \mid (\nu z)\,(!\,d\to z \mid !\,[\![P]\!]_{\mathsf{s}}))) & \sim_{\mathsf{c}}\ (\alpha,x) \\
y(s,d).\,((\nu\beta)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid !\,\beta\to s) \mid (\nu z)\,(!\,d\to z \mid !\,[\![P]\!]_{\mathsf{s}})) & \triangleq\ [\![Q\widehat{\beta}\,[y]\,\widehat{z}P]\!]_{\mathsf{s}}
\end{array}
$$

$(\mathit{exp\text{-}imp})$: $(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \to Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R)$.

$$
\begin{array}{ll}
[\![(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R)]\!]_{\mathsf{s}} \triangleq (\nu\alpha x)\,(!\,[\![\widehat{y}P\widehat{\beta}{\cdot}\alpha]\!]_{\mathsf{s}} \mid !\,\alpha\to x \mid !\,[\![Q\widehat{\gamma}\,[x]\,\widehat{z}R]\!]_{\mathsf{s}}) & {}_{\mathsf{c}}\!\sqsupseteq \\
(\nu\alpha x)\,([\![\widehat{y}P\widehat{\beta}{\cdot}\alpha]\!]_{\mathsf{s}} \mid \alpha\to x \mid [\![Q\widehat{\gamma}\,[x]\,\widehat{z}R]\!]_{\mathsf{s}}) & \triangleq \\
(\nu\alpha x)\,((\nu y\beta)\,(!\,[\![P]\!]_{\mathsf{s}} \mid \overline{\alpha}\langle y,\beta\rangle) \mid \alpha\to x \mid & \\
\qquad x(s,d).\,((\nu\gamma)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid !\,\gamma\to s) \mid (\nu z)\,(!\,d\to z \mid !\,[\![R]\!]_{\mathsf{s}}))) & \to_{\pi}^{+}\ (\alpha,x) \\
(\nu y\beta\gamma z)\,(!\,[\![P]\!]_{\mathsf{s}} \mid !\,[\![Q]\!]_{\mathsf{s}} \mid !\,\gamma\to y \mid !\,\beta\to z \mid !\,[\![R]\!]_{\mathsf{s}}) & \equiv \\
(\nu\gamma y)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid !\,\gamma\to y \mid (\nu\beta z)\,(!\,[\![P]\!]_{\mathsf{s}} \mid !\,\beta\to z \mid !\,[\![R]\!]_{\mathsf{s}})) & \sim_{\mathsf{c}}\ (3.7\,(1a)) \\
(\nu\gamma y)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid !\,\gamma\to y \mid !\,(\nu\beta z)\,(!\,[\![P]\!]_{\mathsf{s}} \mid !\,\beta\to z \mid !\,[\![R]\!]_{\mathsf{s}})) & \triangleq\ [\![Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R)]\!]_{\mathsf{s}}
\end{array}
$$

For $(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \to (Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R$ the proof is similar:

$$
\begin{array}{ll}
(\nu y\beta\gamma z)\,(!\,[\![P]\!]_{\mathsf{s}} \mid !\,[\![Q]\!]_{\mathsf{s}} \mid !\,\gamma\to y \mid !\,\beta\to z \mid !\,[\![R]\!]_{\mathsf{s}}) & \equiv,\sim_{\mathsf{c}} \\
(\nu\beta z)\,(!\,(\nu\gamma y)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid !\,\gamma\to y \mid !\,[\![P]\!]_{\mathsf{s}}) \mid !\,\beta\to z \mid !\,[\![R]\!]_{\mathsf{s}}) & \triangleq\ [\![(Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R]\!]_{\mathsf{s}}
\end{array}
$$

Activation rules: Trivial.

Left propagation:

$(\dagger\not\dagger)$: $\langle y{\cdot}\alpha\rangle\widehat{\alpha} \not\dagger \widehat{x}P \to \langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}P$. Trivial.

$(\mathit{cap}\not\dagger)$: $\langle y{\cdot}\beta\rangle\widehat{\alpha} \not\dagger \widehat{x}P \to \langle y{\cdot}\beta\rangle$, $\beta \neq \alpha$.

$$
\begin{array}{l}
[\![\langle y{\cdot}\beta\rangle\widehat{\alpha} \not\dagger \widehat{x}P]\!]_{\mathsf{s}} \triangleq (\nu\alpha x)\,(!\,y(w).\,\overline{\beta}\langle w\rangle \mid !\,\alpha\to x \mid !\,[\![P]\!]_{\mathsf{s}}) \ \equiv\ (\beta \neq \alpha) \\
\qquad !\,y(w).\,\overline{\beta}\langle w\rangle \mid (\nu\alpha x)\,(!\,\alpha\to x \mid !\,[\![P]\!]_{\mathsf{s}}) \ {}_{\mathsf{c}}\!\sqsupseteq\ y(w).\,\overline{\beta}\langle w\rangle \ \triangleq\ [\![\langle y{\cdot}\beta\rangle]\!]_{\mathsf{s}}
\end{array}
$$

Notice that, in case P does not contain α, $(\nu\alpha x)\,(!\,\alpha\to x \mid [\![P]\!]_{\mathsf{s}}) \equiv (\nu\alpha x)\,(!\,\alpha\to x) \mid [\![P]\!]_{\mathsf{s}}$; it is this what forces the theorem itself to be stated using ${}_{\mathsf{c}}\!\sqsupseteq$.

$(\mathit{exp\text{-}outs}\not\dagger)$: $(\widehat{y}Q\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \not\dagger \widehat{x}P \to (\widehat{y}(Q\widehat{\alpha} \not\dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{x}P$, γ *fresh*.

$$
\begin{array}{ll}
[\![(\widehat{y}Q\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \not\dagger \widehat{x}P]\!]_{\mathsf{s}} \triangleq (\nu\alpha x)\,(!\,(\nu y\beta)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid \overline{\alpha}\langle y,\beta\rangle) \mid !\,\alpha\to x \mid !\,[\![P]\!]_{\mathsf{s}}) & \sim_{\mathsf{c}}\ (3.7\,(1b)) \\
!\,(\nu\alpha x)\,((\nu y\beta)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid \overline{\alpha}\langle y,\beta\rangle) \mid !\,\alpha\to x \mid !\,[\![P]\!]_{\mathsf{s}}) & \sim_{\mathsf{c}}\ (3.7\,(2b)) \\
!\,(\nu\gamma x)\,((\nu y\beta)\,((\nu\alpha x)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid !\,\alpha\to x \mid !\,[\![P]\!]_{\mathsf{s}}) \mid \overline{\gamma}\langle y,\beta\rangle) \mid !\,\gamma\to x \mid !\,[\![P]\!]_{\mathsf{s}}) & \sim_{\mathsf{c}}\ (3.7\,(1b)) \\
(\nu\gamma x)\,(!\,(\nu y\beta)\,((\nu\alpha x)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid !\,\alpha\to x \mid !\,[\![P]\!]_{\mathsf{s}}) \mid \overline{\gamma}\langle y,\beta\rangle) \mid !\,\gamma\to x \mid !\,[\![P]\!]_{\mathsf{s}}) & \sim_{\mathsf{c}}\ (3.7\,(1a)) \\
(\nu\gamma x)\,(!\,(\nu y\beta)\,(!\,(\nu\alpha x)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid !\,\alpha\to x \mid !\,[\![P]\!]_{\mathsf{s}}) \mid \overline{\gamma}\langle y,\beta\rangle) \mid !\,\gamma\to x \mid !\,[\![P]\!]_{\mathsf{s}}) & \triangleq \\
[\![(\widehat{y}(Q\widehat{\alpha} \not\dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{x}P]\!]_{\mathsf{s}} &
\end{array}
$$

$(\mathit{exp\text{-}ins}\not\dagger)$: $(\widehat{y}Q\widehat{\beta}{\cdot}\gamma)\widehat{\alpha} \not\dagger \widehat{x}P \to \widehat{y}(Q\widehat{\alpha} \not\dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma, \gamma \neq \alpha$.

$$
\begin{array}{lll}
[\![(\widehat{y}Q\widehat{\beta}{\cdot}\gamma)\widehat{\alpha} \not\dagger \widehat{x}P]\!]_{\mathsf{s}} \triangleq & (\nu\alpha x)\,(!\,(\nu y\beta)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid \overline{\gamma}\langle y,\beta\rangle) \mid !\,\alpha\to x \mid !\,[\![P]\!]_{\mathsf{s}}) & \sim_{\mathsf{c}}\ (3.7\,(1b)) \\
& !\,(\nu\alpha x)\,((\nu y\beta)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid \overline{\gamma}\langle y,\beta\rangle) \mid !\,\alpha\to x \mid !\,[\![P]\!]_{\mathsf{s}}) & \equiv \\
& !\,(\nu y\beta)\,((\nu\alpha x)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid !\,\alpha\to x \mid !\,[\![P]\!]_{\mathsf{s}}) \mid \overline{\gamma}\langle y,\beta\rangle) & {}_{\mathsf{c}}\!\sqsupseteq \\
& (\nu y\beta)\,((\nu\alpha x)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid !\,\alpha\to x \mid !\,[\![P]\!]_{\mathsf{s}}) \mid \overline{\gamma}\langle y,\beta\rangle) & \sim_{\mathsf{c}}\ (3.7\,(1a)) \\
& (\nu y\beta)\,(!\,(\nu\alpha x)\,(!\,[\![Q]\!]_{\mathsf{s}} \mid !\,\alpha\to x \mid !\,[\![P]\!]_{\mathsf{s}}) \mid \overline{\gamma}\langle y,\beta\rangle) & \triangleq \\
& [\![\widehat{y}(Q\widehat{\alpha} \not\dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma]\!]_{\mathsf{s}} &
\end{array}
$$

$(imp\not\nearrow)$**:** Excluded from $\rightarrow_{\mathrm{H}}$.

$(cut\not\nearrow)$**:** $(Q\widehat{\beta}\dagger\widehat{y}R)\widehat{\alpha}\not\nearrow\widehat{x}P \rightarrow (Q\widehat{\alpha}\not\nearrow\widehat{x}P)\widehat{\beta}\dagger\widehat{y}(R\widehat{\alpha}\not\nearrow\widehat{x}P)$.

$$\begin{array}{ll}
[\![(Q\widehat{\beta}\dagger\widehat{y}R)\widehat{\alpha}\not\nearrow\widehat{x}P]\!]_{\mathrm{s}} \triangleq (\nu\alpha x)\,(!\,(\nu\beta y)\,(!\,[\![Q]\!]_{\mathrm{s}}\,|\,!\,\beta{\rightarrow}y\,|\,!\,[\![R]\!]_{\mathrm{s}})\,|\,!\,\alpha{\rightarrow}x\,|\,!\,[\![P]\!]_{\mathrm{s}}) & \sim_{\mathsf{C}}\ (3.7\,(1a))\\
!\,(\nu\alpha x)\,((\nu\beta y)\,(!\,[\![Q]\!]_{\mathrm{s}}\,|\,!\,\beta{\rightarrow}y\,|\,!\,[\![R]\!]_{\mathrm{s}})\,|\,!\,\alpha{\rightarrow}x\,|\,!\,[\![P]\!]_{\mathrm{s}}) & \sim_{\mathsf{C}}\ (3.7\,(2a))\\
!\,(\nu\beta y)\,((\nu\alpha x)\,(!\,[\![Q]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,!\,[\![P]\!]_{\mathrm{s}})\,|\,!\,\beta{\rightarrow}y\,|\,(\nu\alpha x)\,(!\,[\![R]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,!\,[\![P]\!]_{\mathrm{s}})) & \sim_{\mathsf{C}}\ (3.7\,(1a))\\
(\nu\beta y)\,(!\,(\nu\alpha x)\,(!\,[\![Q]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,!\,[\![R]\!]_{\mathrm{s}})\,|\,!\,\beta{\rightarrow}y\,|\,!\,(\nu\alpha x)\,(!\,[\![R]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,!\,[\![P]\!]_{\mathrm{s}})) & \triangleq\\
[\![(Q\widehat{\alpha}\not\nearrow\widehat{x}P)\widehat{\beta}\dagger\widehat{y}(R\widehat{\alpha}\not\nearrow\widehat{x}P)]\!]_{\mathrm{s}} &
\end{array}$$

Right propagation:

$(\not\nwarrow\dagger)$**:** $P\widehat{\alpha}\not\nwarrow\widehat{x}\langle x{\cdot}\beta\rangle \rightarrow P\widehat{\alpha}\dagger\widehat{x}\langle x{\cdot}\beta\rangle$, α not introduced in P. Trivial.

$(\not\nwarrow cap)$**:** Then $P\widehat{\alpha}\not\nwarrow\widehat{x}\langle y{\cdot}\beta\rangle \rightarrow \langle y{\cdot}\beta\rangle, y\neq x$.

$$\begin{array}{l}
[\![P\widehat{\alpha}\not\nwarrow\widehat{x}\langle y{\cdot}\beta\rangle]\!]_{\mathrm{s}} \triangleq (\nu\alpha x)\,(!\,[\![P]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,!\,y(w).\,\overline{\beta}\langle w\rangle) \ \equiv\ (y\neq x)\\
\qquad\qquad\qquad (\nu\alpha x)\,(!\,[\![P]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x)\,|\,!\,y(w).\,\overline{\beta}\langle w\rangle \ {}_{\mathsf{C}}\!\sqsupseteq\ y(w).\,\overline{\beta}\langle w\rangle \ \triangleq\ [\![\langle y{\cdot}\beta\rangle]\!]_{\mathrm{s}}
\end{array}$$

Note again the use of ${}_{\mathsf{C}}\!\sqsupseteq$.

$(\not\nwarrow exp)$**:** Then $P\widehat{\alpha}\not\nwarrow\widehat{x}(\widehat{y}Q\widehat{\beta}{\cdot}\gamma) \rightarrow \widehat{y}(P\widehat{\alpha}\not\nwarrow\widehat{x}Q)\widehat{\beta}{\cdot}\gamma$.

$$\begin{array}{ll}
[\![P\widehat{\alpha}\not\nwarrow\widehat{x}(\widehat{y}Q\widehat{\beta}{\cdot}\gamma)]\!]_{\mathrm{s}} \triangleq (\nu\alpha x)\,(!\,[\![P]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,!\,(\nu y\beta)\,(!\,[\![Q]\!]_{\mathrm{s}}\,|\,\overline{\gamma}\langle y,\beta\rangle)) & \sim_{\mathsf{C}}\ (3.7\,(1b))\\
!\,(\nu\alpha x)\,(!\,[\![P]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,(\nu y\beta)\,(!\,[\![Q]\!]_{\mathrm{s}}\,|\,\overline{\gamma}\langle y,\beta\rangle)) & \equiv\\
!\,(\nu y\beta)\,((\nu\alpha x)\,(!\,[\![P]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,!\,[\![Q]\!]_{\mathrm{s}})\,|\,\overline{\gamma}\langle y,\beta\rangle) & \sim_{\mathsf{C}}\ (3.7\,(1b))\\
(\nu y\beta)\,(!\,(\nu\alpha x)\,(!\,[\![P]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,!\,[\![Q]\!]_{\mathrm{s}})\,|\,\overline{\gamma}\langle y,\beta\rangle) & \triangleq\ [\![\widehat{y}(P\widehat{\alpha}\not\nwarrow\widehat{x}Q)\widehat{\beta}{\cdot}\gamma]\!]_{\mathrm{s}}
\end{array}$$

$(\not\nwarrow imp\text{-}outs), (\not\nwarrow imp\text{-}ins)$**:** Excluded from $\rightarrow_{\mathrm{H}}$.

$(\not\nwarrow cut)$**:** Then $P\widehat{\alpha}\not\nwarrow\widehat{x}(Q\widehat{\beta}\dagger\widehat{y}R) \rightarrow_{\mathcal{X}} (P\widehat{\alpha}\not\nwarrow\widehat{x}Q)\widehat{\beta}\dagger\widehat{y}(P\widehat{\alpha}\not\nwarrow\widehat{x}R)$.

$$\begin{array}{ll}
[\![P\widehat{\alpha}\not\nwarrow\widehat{x}(Q\widehat{\beta}\dagger\widehat{y}R)]\!]_{\mathrm{s}} \triangleq (\nu\alpha x)\,(!\,[\![P]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,!\,(\nu\beta y)\,(!\,[\![Q]\!]_{\mathrm{s}}\,|\,!\,\beta{\rightarrow}y\,|\,!\,[\![R]\!]_{\mathrm{s}})) & \sim_{\mathsf{C}}\ (3.7\,(1b))\\
!\,(\nu\alpha x)\,(!\,[\![P]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,(\nu\beta y)\,(!\,[\![Q]\!]_{\mathrm{s}}\,|\,!\,\beta{\rightarrow}y\,|\,!\,[\![R]\!]_{\mathrm{s}})) & \sim_{\mathsf{C}}\ (3.7\,(3a))\\
!\,(\nu\beta y)\,((\nu\alpha x)\,(!\,[\![P]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,!\,[\![Q]\!]_{\mathrm{s}})\,|\,!\,\beta{\rightarrow}y\,|\,(\nu\alpha x)\,(!\,[\![P]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,!\,[\![R]\!]_{\mathrm{s}})) & \sim_{\mathsf{C}}\ (3.7\,(1b))\\
(\nu\beta y)\,(!\,(\nu\alpha x)\,(!\,[\![P]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,!\,[\![Q]\!]_{\mathrm{s}})\,|\,!\,\beta{\rightarrow}y\,|\,!\,(\nu\alpha x)\,(!\,[\![P]\!]_{\mathrm{s}}\,|\,!\,\alpha{\rightarrow}x\,|\,!\,[\![R]\!]_{\mathrm{s}})) & \triangleq\\
[\![(P\widehat{\alpha}\not\nwarrow\widehat{x}Q)\widehat{\beta}\dagger\widehat{y}(P\widehat{\alpha}\not\nwarrow\widehat{x}R)]\!]_{\mathrm{s}} &
\end{array}$$

Contextual rules:

$P\rightarrow Q \Rightarrow \widehat{x}P\widehat{\alpha}{\cdot}\beta \rightarrow \widehat{x}Q\widehat{\alpha}{\cdot}\beta$**:**

$$[\![\widehat{x}P\widehat{\alpha}{\cdot}\beta]\!]_{\mathrm{s}} \triangleq (\nu x\alpha)\,(!\,[\![P]\!]_{\mathrm{s}}\,|\,\overline{\beta}\langle x,\alpha\rangle) \ {}_{\mathsf{C}}\!\sqsupseteq\ (IH)\ (\nu x\alpha)\,(!\,[\![Q]\!]_{\mathrm{s}}\,|\,\overline{\beta}\langle x,\alpha\rangle) \triangleq [\![\widehat{x}Q\widehat{\alpha}{\cdot}\beta]\!]_{\mathrm{s}}$$

$P\rightarrow Q \Rightarrow P\widehat{\alpha}\dagger\widehat{y}R \rightarrow Q\widehat{\alpha}\dagger\widehat{y}R,\ R\widehat{\alpha}\dagger\widehat{y}P \rightarrow R\widehat{\alpha}\dagger\widehat{y}Q$**:** By induction.

$P\rightarrow Q \Rightarrow P\widehat{\alpha}\,[x]\,\widehat{y}R \rightarrow Q\widehat{\alpha}\,[x]\,\widehat{y}R,\ R\widehat{\alpha}\,[x]\,\widehat{y}P \rightarrow R\widehat{\alpha}\,[x]\,\widehat{y}Q$**:** Excluded from $\rightarrow_{\mathrm{H}}$.

$P\rightarrow Q\ \&\ Q\rightarrow R \Rightarrow P\rightarrow R$**:** By induction.

$P\rightarrow Q \Rightarrow R\widehat{\alpha}\dagger\widehat{y}P \rightarrow R\widehat{\alpha}\dagger\widehat{y}Q$**:** By induction. □

Notice that, in the proof above, the only place where reduction plays a role is in the logical rules; all other steps are dealt with by the congruence rules, contextual equivalence and/or induction.

Observe that the image of $\mathcal{X}$ in π, being built without using 'choice', has no notion of *erasure* of processes; the *cut* $P\widehat{\alpha}\dagger\widehat{x}Q$, with α not in P and x not in Q, in $\mathcal{X}$ runs via erasure to either P or

Q, and reducing it decreases the set of reachable normal forms; but:

$$\begin{array}{rcl}\llbracket P\widehat{\alpha} \dagger \widehat{x}Q \rrbracket_{\mathsf{s}} & \triangleq & (\nu\alpha x)\,(!\,\llbracket P \rrbracket_{\mathsf{s}} \mid !\,\alpha{\rightarrow}x \mid !\,\llbracket Q \rrbracket_{\mathsf{s}}) \\ & \equiv & !\,\llbracket P \rrbracket_{\mathsf{s}} \mid (\nu\alpha x)\,(!\,\alpha{\rightarrow}x) \mid !\,\llbracket Q \rrbracket_{\mathsf{s}} \;\equiv\; !\,\llbracket P \rrbracket_{\mathsf{s}} \mid !\,\llbracket Q \rrbracket_{\mathsf{s}}\end{array}$$

which, evidently, has more behaviour than both $\llbracket P \rrbracket_{\mathsf{s}}$ and $\llbracket Q \rrbracket_{\mathsf{s}}$. So for any $\mathcal{X}$-term P, $\llbracket P \rrbracket_{\mathsf{s}}$ essentially 'contains' all normal forms of P in parallel; restricting to either (confluent) call-by-name or call-by-value reductions, this feature will disappear.

The result presented in [33] is stronger, but only achieved for Call-by-Value $\lambda\mu$, and at the price of a very intricate translation that depends on types; since $\lambda\mu$ is confluent, normal forms are unique. The result as presented in [20] is achieved for outermost-reduction in $\overline{\lambda}\mu\tilde{\mu}$; it strongly depends on recursion, and is not compositional.

Example 4.5. The encoding of $\llbracket \widehat{z}((\widehat{y}\langle y{\cdot}\delta\rangle\widehat{\eta}{\cdot}\alpha)\widehat{\alpha}\ [z]\ \widehat{v}\langle v{\cdot}\delta\rangle)\widehat{\delta}{\cdot}\gamma \rrbracket_{\mathsf{s}}$, *i.e.* the witness of Peirce's law, becomes:

$$\begin{array}{l}(\nu z\delta)\,(z(s,d).\,((\nu\alpha)\,!\,((\nu y\eta)\,(!\,y(w).\,\overline{\delta}\langle w\rangle \mid \overline{\alpha}\langle y,\eta\rangle) \mid !\,\alpha{\rightarrow}s) \mid \\ \qquad\qquad\qquad\qquad (\nu v)\,(!\,d{\rightarrow}v \mid !\,v(w).\,\overline{\delta}\langle w\rangle))) \mid \overline{\gamma}\langle z,\delta\rangle)\end{array}$$

That this process is a witness of $((A{\to}B){\to}A){\to}A$ is a straightforward application of Theorem 6.7.

Notice that the second reduction in Example 2.8 propagates into an *import*, so by head reduction is limited to:

$$\begin{array}{lcl}(\widehat{z}P\widehat{\delta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{u}(\langle u{\cdot}\beta\rangle\widehat{\beta} \dagger \widehat{y}(Q\widehat{\tau}\,[y]\,\widehat{w}R)) & \to & (\nwarrow a, \nwarrow \mathit{cut}) \\ ((\widehat{z}P\widehat{\delta}{\cdot}\gamma)\widehat{\gamma} \nwarrow \widehat{u}\langle u{\cdot}\beta\rangle)\widehat{\beta} \dagger \widehat{y}((\widehat{z}P\widehat{\delta}{\cdot}\gamma)\widehat{\gamma} \nwarrow \widehat{u}(Q\widehat{\tau}\,[y]\,\widehat{w}R)) & \to & (\nwarrow d, \mathit{exp}) \\ (\widehat{z}P\widehat{\delta}{\cdot}\beta)\widehat{\beta} \dagger \widehat{y}((\widehat{z}P\widehat{\delta}{\cdot}\gamma)\widehat{\gamma} \nwarrow \widehat{u}(Q\widehat{\tau}\,[y]\,\widehat{w}R)) & \to & (\nwarrow a) \\ (\widehat{z}P\widehat{\delta}{\cdot}\beta)\widehat{\beta} \nwarrow \widehat{y}((\widehat{z}P\widehat{\delta}{\cdot}\gamma)\widehat{\gamma} \nwarrow \widehat{u}(Q\widehat{\tau}\,[y]\,\widehat{w}R)) & & \end{array}$$

where the last term is in head-normal form. Since the first reduction in Example 2.8 is also a head reduction, this shows that head reduction is not confluent.

This reduction is modelled in π by:

$$\begin{array}{ll}\llbracket (\widehat{z}P\widehat{\delta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{u}(\langle u{\cdot}\beta\rangle\widehat{\beta} \dagger \widehat{y}(Q\widehat{\tau}\,[y]\,\widehat{w}R)) \rrbracket_{\mathsf{s}} & \triangleq \\ (\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}{\cdot}\gamma \rrbracket_{\mathsf{s}} \mid !\,\gamma{\rightarrow}u \mid !\,\llbracket \langle u{\cdot}\beta\rangle\widehat{\beta} \dagger \widehat{y}(Q\widehat{\tau}\,[y]\,\widehat{w}R) \rrbracket_{\mathsf{s}}) & \triangleq \\ (\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}{\cdot}\gamma \rrbracket_{\mathsf{s}} \mid !\,\gamma{\rightarrow}u \mid !\,(\nu\beta y)\,(!\,u(w).\,\overline{\beta}\langle w\rangle \mid !\,\beta{\rightarrow}y \mid !\,\llbracket Q\widehat{\tau}\,[y]\,\widehat{w}R \rrbracket_{\mathsf{s}})) & \sim_{\mathsf{c}}, \triangleq\ (3.7) \\ (\nu\beta y)\,(!\,(\nu\gamma u)\,(!\,(\nu z\delta)\,(!\,\llbracket P \rrbracket_{\mathsf{s}} \mid \overline{\gamma}\langle z,\delta\rangle) \mid !\,\gamma{\rightarrow}u \mid !\,\langle u{\cdot}\beta\rangle) \mid !\,\beta{\rightarrow}y \mid & \\ \qquad\qquad !\,(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}{\cdot}\gamma \rrbracket_{\mathsf{s}} \mid !\,\gamma{\rightarrow}u \mid !\,\llbracket Q\widehat{\tau}\,[y]\,\widehat{w}R \rrbracket_{\mathsf{s}})) & \sim_{\mathsf{c}}\ (\gamma,u) \\ (\nu\beta y)\,(!\,(\nu z\delta)\,(!\,\llbracket P \rrbracket_{\mathsf{s}} \mid \overline{\beta}\langle z,\delta\rangle) \mid !\,\beta{\rightarrow}y \mid !\,(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}{\cdot}\gamma \rrbracket_{\mathsf{s}} \mid !\,\gamma{\rightarrow}u \mid !\,\llbracket Q\widehat{\tau}\,[y]\,\widehat{w}R \rrbracket_{\mathsf{s}})) & \triangleq \\ \llbracket (\widehat{z}P\widehat{\delta}{\cdot}\beta)\widehat{\beta} \dagger \widehat{y}((\widehat{z}P\widehat{\delta}{\cdot}\gamma)\widehat{\gamma} \nwarrow \widehat{u}(Q\widehat{\tau}\,[y]\,\widehat{w}R)) \rrbracket_{\mathsf{s}} & \triangleq \\ \llbracket (\widehat{z}P\widehat{\delta}{\cdot}\beta)\widehat{\beta} \nwarrow \widehat{y}((\widehat{z}P\widehat{\delta}{\cdot}\gamma)\widehat{\gamma} \nwarrow \widehat{u}(Q\widehat{\tau}\,[y]\,\widehat{w}R)) \rrbracket_{\mathsf{s}} & \end{array}$$

Consider now the third reduction of $(\widehat{z}P\widehat{\delta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{u}(Q\widehat{\tau}\,[u]\,\widehat{w}R)$, where $P = \langle z{\cdot}\delta\rangle$, $Q = \langle u{\cdot}\tau\rangle$ and $R = \langle w{\cdot}\sigma\rangle$ (notice that, in Example 2.8, $Q = \langle v{\cdot}\tau\rangle$), so u is not introduced in $Q\widehat{\tau}\,[u]\,\widehat{w}R$. Then the head reduction on this term runs only as follows:

$$(\widehat{z}P\widehat{\delta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{u}(Q\widehat{\tau}\,[u]\,\widehat{w}R)\ \to (\nwarrow a)\ \ (\widehat{z}P\widehat{\delta}{\cdot}\gamma)\widehat{\gamma} \nwarrow \widehat{u}(Q\widehat{\tau}\,[u]\,\widehat{w}R)$$

Since activated *cuts* are interpreted in the same way as inactive *cuts*, this reduction is modelled in the π-calculus by equality.

Notice that, since

$$\begin{array}{l} (\nu\beta y)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\beta\rrbracket_{\mathrm{S}} \mid !\,\beta{\rightarrow}y \mid !\,(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{S}} \mid !\,\gamma{\rightarrow}u \mid !\,\llbracket Q\widehat{\tau}\,[y]\,\widehat{w}R\rrbracket_{\mathrm{S}})) \quad \triangleq \\ (\nu\beta y)\,(!\,(\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{S}} \mid \overline{\beta}\langle z,\delta\rangle) \mid !\,\beta{\rightarrow}y \mid !\,(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{S}} \mid !\,\gamma{\rightarrow}u \mid \\ \qquad\qquad !\,y(s,d).\,((\nu\tau)\,(!\,\llbracket Q\rrbracket_{\mathrm{S}} \mid !\,\tau{\rightarrow}s) \mid (\nu w)\,(!\,d{\rightarrow}w \mid !\,\llbracket R\rrbracket_{\mathrm{S}})))) \end{array}$$

there is still a communication possible over γ and y and therefore the interpretation of

$$(\widehat{z}P\widehat{\delta}\cdot\beta)\widehat{\beta} \nwarrow \widehat{y}((\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \nwarrow \widehat{u}(Q\widehat{\tau}\,[y]\,\widehat{w}R))$$

as appears above *can* reduce:

$$\begin{array}{ll} \llbracket (\widehat{z}P\widehat{\delta}\cdot\beta)\widehat{\beta} \nwarrow \widehat{y}((\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \nwarrow \widehat{u}(Q\widehat{\tau}\,[y]\,\widehat{w}R))\rrbracket_{\mathrm{S}} & \triangleq \\ (\nu\beta y)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\beta\rrbracket_{\mathrm{S}} \mid !\,\beta{\rightarrow}y \mid !\,(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{S}} \mid !\,\gamma{\rightarrow}u \mid !\,\llbracket Q\widehat{\tau}\,[y]\,\widehat{w}R\rrbracket_{\mathrm{S}})) & \triangleq \\ (\nu\beta y)\,(!\,(\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{S}} \mid \overline{\beta}\langle z,\delta\rangle) \mid !\,\beta{\rightarrow}y \mid !\,(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{S}} \mid !\,\gamma{\rightarrow}u \mid & \\ \qquad\qquad !\,y(s,d).\,((\nu\tau)\,(!\,\llbracket Q\rrbracket_{\mathrm{S}} \mid !\,\tau{\rightarrow}s) \mid (\nu w)\,(!\,d{\rightarrow}w \mid !\,\llbracket R\rrbracket_{\mathrm{S}})))) & {}_{\mathrm{c}}\sqsupseteq \\ (\nu\beta y)\,((\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{S}} \mid \overline{\beta}\langle z,\delta\rangle) \mid \beta{\rightarrow}y \mid (\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{S}} \mid !\,\gamma{\rightarrow}u \mid & \\ \qquad\qquad y(s,d).\,((\nu\tau)\,(!\,\llbracket Q\rrbracket_{\mathrm{S}} \mid !\,\tau{\rightarrow}s) \mid (\nu w)\,(!\,d{\rightarrow}w \mid !\,\llbracket R\rrbracket_{\mathrm{S}})))) & \rightarrow_{\pi}\ (\beta,y) \\ (\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{S}} \mid (\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{S}} \mid !\,\gamma{\rightarrow}u \mid & \\ \qquad\qquad\qquad (\nu\tau)\,(!\,\llbracket Q\rrbracket_{\mathrm{S}} \mid !\,\tau{\rightarrow}z) \mid (\nu w)\,(!\,\delta{\rightarrow}w \mid \llbracket R\rrbracket_{\mathrm{S}}))) & \end{array}$$

which removes the *input* (and allows computation inside Q and R to be modelled, if any were present); this implies that the simple encoding captures more than just head reduction. This is essentially caused by the fact that we encode all *cuts* in the same way, thereby modeling, in the interpretation, that activated *cuts* propagate over activated *cuts*, as activated propagate over unactivated.

5. EMBEDDING $\mathcal{X}$'S REDUCTION IN FULL

In this section, we define an encoding from terms in $\mathcal{X}$ onto processes in π that fully respects reduction in $\mathcal{X}$, as a variant of the encoding presented above. In the approach of $\llbracket\cdot\rrbracket_{\mathrm{S}}$, the *import* $P\widehat{\alpha}\,[x]\,\widehat{y}Q$ gets expressed using $(\nu\alpha)\,(!\,\llbracket P\rrbracket_{\mathrm{S}} \mid !\,\alpha{\rightarrow}s)$ and $(\nu y)\,(!\,d{\rightarrow}y \mid !\,\llbracket Q\rrbracket_{\mathrm{S}})$. However, the variables s and d appear *only* in the redirections, not in $\llbracket P\rrbracket_{\mathrm{S}}$ or $\llbracket Q\rrbracket_{\mathrm{S}}$, so these two processes appear unnecessarily under *input* in the encoding $\llbracket\cdot\rrbracket_{\mathrm{S}}$. This is what the new encoding $\llbracket\cdot\rrbracket_{\mathrm{F}}$ fixes: we build what we call a *communication cell* in $x(s,d).\,(!\,\alpha{\rightarrow}s \mid !\,d{\rightarrow}y)$, which deals with the redirections of the received mediator's interface, which we put in parallel with the (replicated) encodings of $\llbracket P\rrbracket_{\mathrm{F}}$ and $\llbracket Q\rrbracket_{\mathrm{F}}$, creating the process

$$!\,\llbracket P\rrbracket_{\mathrm{F}} \mid x(s,d).\,(!\,\alpha{\rightarrow}s \mid !\,d{\rightarrow}y) \mid !\,\llbracket Q\rrbracket_{\mathrm{F}}$$

We only need to express that the names α and y are not visible from outside this process; notice that, by construction α occurs only in $\llbracket P\rrbracket_{\mathrm{F}}$, and and y only in $\llbracket Q\rrbracket_{\mathrm{F}}$.

So we define:

Definition 5.1 (Full interpretation of $\mathcal{X}$ into π).

$$\begin{array}{rcll} \llbracket \langle x{\cdot}\alpha\rangle \rrbracket_{\mathrm{F}} & = & x(w).\,\overline{\alpha}\langle w\rangle & \\ \llbracket \widehat{y}Q\widehat{\beta}\cdot\alpha\rrbracket_{\mathrm{F}} & = & (\nu y\beta)\,(!\,\llbracket Q\rrbracket_{\mathrm{F}} \mid \overline{\alpha}\langle y,\beta\rangle) & \\ \llbracket P\widehat{\alpha}\,[x]\,\widehat{y}Q\rrbracket_{\mathrm{F}} & = & (\nu\alpha y)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid x(v,d).\,(!\,\alpha{\rightarrow}v \mid !\,d{\rightarrow}y) \mid !\,\llbracket Q\rrbracket_{\mathrm{F}}) & \\ \llbracket P\widehat{\alpha}\dagger\widehat{x}Q\rrbracket_{\mathrm{F}} & = & (\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha{\rightarrow}x \mid !\,\llbracket Q\rrbracket_{\mathrm{F}}) & = \llbracket P\widehat{\alpha} \nearrow \widehat{x}Q\rrbracket_{\mathrm{F}} = \llbracket P\widehat{\alpha} \nwarrow \widehat{x}Q\rrbracket_{\mathrm{F}} \end{array}$$

Notice that (as in Def. 4.1) all *cuts* are interpreted the same way.

Example 5.2. The encoding of $\llbracket \widehat{z}((\widehat{y}\langle y{\cdot}\delta\rangle\widehat{\eta}{\cdot}\alpha)\widehat{\alpha}\ [z]\ \widehat{v}\langle v{\cdot}\delta\rangle)\widehat{\delta}{\cdot}\gamma \rrbracket_{\mathrm{F}}$, *i.e.* the witness of Peirce's law, becomes:

$$(\nu z\delta)\,(!\,(\nu\alpha v)\,(!\,(\nu y\eta)\,(!\,y(w).\overline{\delta}\langle w\rangle \mid \overline{\alpha}\langle y,\eta\rangle) \mid z(v,d).\,(!\,\alpha{\rightarrow}v \mid !\,d{\rightarrow}v) \mid !\,v(w).\overline{\delta}\langle w\rangle) \mid \overline{\gamma}\langle z,\delta\rangle)$$

That this process is a witness of $\vdash_\pi\ \gamma{:}((A{\to}B){\to}A){\to}A$ is a straightforward application of Theorem 6.7.

As above, we will show a preservation result for this encoding modulo *contextually equivalence*.

Theorem 5.3. *If $P \to_{\mathcal{X}} Q$, then $\llbracket P \rrbracket_{\mathrm{F}}\ {}_{\mathsf{C}}\!\sqsupseteq\ \llbracket Q \rrbracket_{\mathrm{F}}$.*

Proof. Since the only difference between $\llbracket P \rrbracket_{\mathrm{F}}$ and $\llbracket P \rrbracket_{\mathrm{S}}$ is the interpretation of *imports*, we only need to check the rules involving *imports*; some of these were not considered in the proof of Theorem 4.4, since omitted from $\to_{\mathrm{H}}$.

(imp): $\langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{z}P) \to Q\widehat{\beta}\,[y]\,\widehat{z}P$.

$$\begin{array}{ll}
\llbracket \langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{z}P) \rrbracket_{\mathrm{F}} \triangleq (\nu\alpha x)\,(!\,\llbracket \langle y{\cdot}\alpha\rangle \rrbracket_{\mathrm{F}} \mid !\,\alpha{\rightarrow}x \mid !\,\llbracket Q\widehat{\beta}\,[x]\,\widehat{z}P \rrbracket_{\mathrm{F}}) & {}_{\mathsf{C}}\!\sqsupseteq \\
(\nu\alpha x)\,(\llbracket \langle y{\cdot}\alpha\rangle \rrbracket_{\mathrm{F}} \mid \alpha{\rightarrow}x \mid \llbracket Q\widehat{\beta}\,[x]\,\widehat{z}P \rrbracket_{\mathrm{F}}) & \triangleq \\
(\nu\alpha x)\,(y(w).\overline{\alpha}\langle w\rangle \mid \alpha{\rightarrow}x \mid (\nu\beta z)\,(!\,\llbracket Q \rrbracket_{\mathrm{F}} \mid x(v,d).\,(!\,\beta{\rightarrow}v \mid !\,d{\rightarrow}z) \mid !\,\llbracket P \rrbracket_{\mathrm{F}})) & \sim_{\mathsf{C}}\ (\alpha,x) \\
(\nu\beta z)\,(!\,\llbracket Q \rrbracket_{\mathrm{F}} \mid y(v,d).\,(!\,\beta{\rightarrow}v \mid !\,d{\rightarrow}z) \mid !\,\llbracket P \rrbracket_{\mathrm{F}}) & \triangleq\ \llbracket Q\widehat{\beta}\,[y]\,\widehat{z}P \rrbracket_{\mathrm{F}}
\end{array}$$

$(exp\text{-}imp)$: $(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \to Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R)$.

$$\begin{array}{ll}
\llbracket (\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \rrbracket_{\mathrm{F}} \triangleq (\nu\alpha x)\,(!\,\llbracket \widehat{y}P\widehat{\beta}{\cdot}\alpha \rrbracket_{\mathrm{F}} \mid !\,\alpha{\rightarrow}x \mid !\,\llbracket Q\widehat{\gamma}\,[x]\,\widehat{z}R \rrbracket_{\mathrm{F}}) & {}_{\mathsf{C}}\!\sqsupseteq \\
(\nu\alpha x)\,(\llbracket \widehat{y}P\widehat{\beta}{\cdot}\alpha \rrbracket_{\mathrm{F}} \mid \alpha{\rightarrow}x \mid \llbracket Q\widehat{\gamma}\,[x]\,\widehat{z}R \rrbracket_{\mathrm{F}}) & \triangleq \\
(\nu\alpha x)\,((\nu y\beta)\,(!\,\llbracket P \rrbracket_{\mathrm{F}} \mid \overline{\alpha}\langle y,\beta\rangle) \mid \alpha{\rightarrow}x \mid & \\
\qquad\qquad (\nu\gamma z)\,(!\,\llbracket Q \rrbracket_{\mathrm{F}} \mid x(v,d).\,(!\,\gamma{\rightarrow}v \mid !\,d{\rightarrow}z) \mid !\,\llbracket R \rrbracket_{\mathrm{F}})) & \to_\pi^+\ (\alpha,x) \\
(\nu y\beta\gamma z)\,(!\,\llbracket P \rrbracket_{\mathrm{F}} \mid !\,\llbracket Q \rrbracket_{\mathrm{F}} \mid !\,\gamma{\rightarrow}y \mid !\,\beta{\rightarrow}z \mid !\,\llbracket R \rrbracket_{\mathrm{F}}) & \equiv \\
(\nu\gamma y)\,(!\,\llbracket Q \rrbracket_{\mathrm{F}} \mid !\,\gamma{\rightarrow}y \mid (\nu\beta z)\,(!\,\llbracket P \rrbracket_{\mathrm{F}} \mid !\,\beta{\rightarrow}z \mid !\,\llbracket R \rrbracket_{\mathrm{F}})) & \sim_{\mathsf{C}}\ (3.7\,(1a)) \\
(\nu\gamma y)\,(!\,\llbracket Q \rrbracket_{\mathrm{F}} \mid !\,\gamma{\rightarrow}y \mid !\,(\nu\beta z)\,(!\,\llbracket P \rrbracket_{\mathrm{F}} \mid !\,\beta{\rightarrow}z \mid !\,\llbracket R \rrbracket_{\mathrm{F}})) & \triangleq\ \llbracket Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R) \rrbracket_{\mathrm{F}}
\end{array}$$

As for $\llbracket \cdot \rrbracket_{\mathrm{S}}$, for $(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \to (Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R$ the proof is similar.

$(imp\not\nearrow)$: $(Q\widehat{\beta}\,[z]\,\widehat{y}R)\widehat{\alpha} \not\nearrow \widehat{x}P \to (Q\widehat{\alpha} \not\nearrow \widehat{x}P)\widehat{\beta}\,[z]\,\widehat{y}(R\widehat{\alpha} \not\nearrow \widehat{x}P)$.

$$\begin{array}{ll}
\llbracket (Q\widehat{\beta}\,[z]\,\widehat{y}R)\widehat{\alpha} \not\nearrow \widehat{x}P \rrbracket_{\mathrm{F}} & \triangleq \\
(\nu\alpha x)\,(!\,(\nu\beta y)\,(!\,\llbracket Q \rrbracket_{\mathrm{F}} \mid z(v,d).\,(!\,\beta{\rightarrow}v \mid !\,d{\rightarrow}y) \mid !\,\llbracket R \rrbracket_{\mathrm{F}}) \mid !\,\alpha{\rightarrow}x \mid !\,\llbracket P \rrbracket_{\mathrm{F}}) & \sim_{\mathsf{C}}\ (3.7\,(1b)) \\
!\,(\nu\alpha x)\,((\nu\beta y)\,(!\,\llbracket Q \rrbracket_{\mathrm{F}} \mid z(v,d).\,(!\,\beta{\rightarrow}v \mid !\,d{\rightarrow}y) \mid !\,\llbracket R \rrbracket_{\mathrm{F}}) \mid !\,\alpha{\rightarrow}x \mid !\,\llbracket P \rrbracket_{\mathrm{F}}) & \sim_{\mathsf{C}}\ (3.7\,(2a)) \\
!\,(\nu\beta y)\,((\nu\alpha x)\,(!\,\llbracket Q \rrbracket_{\mathrm{F}} \mid !\,\alpha{\rightarrow}x \mid !\,\llbracket P \rrbracket_{\mathrm{F}}) \mid z(v,d).\,(!\,\beta{\rightarrow}v \mid !\,d{\rightarrow}y) \mid & \\
\qquad\qquad (\nu\alpha x)\,(!\,\llbracket R \rrbracket_{\mathrm{F}} \mid !\,\alpha{\rightarrow}x \mid !\,\llbracket P \rrbracket_{\mathrm{F}})) & {}_{\mathsf{C}}\!\sqsupseteq \\
(\nu\beta y)\,((\nu\alpha x)\,(!\,\llbracket Q \rrbracket_{\mathrm{F}} \mid !\,\alpha{\rightarrow}x \mid !\,\llbracket P \rrbracket_{\mathrm{F}}) \mid z(v,d).\,(!\,\beta{\rightarrow}v \mid !\,d{\rightarrow}y) \mid & \\
\qquad\qquad (\nu\alpha x)\,(!\,\llbracket R \rrbracket_{\mathrm{F}} \mid !\,\alpha{\rightarrow}x \mid !\,\llbracket P \rrbracket_{\mathrm{F}})) & \equiv\ (3.7\,(1a)) \\
(\nu\beta y)\,(!\,(\nu\alpha x)\,(!\,\llbracket Q \rrbracket_{\mathrm{F}} \mid !\,\alpha{\rightarrow}x \mid !\,\llbracket R \rrbracket_{\mathrm{F}}) \mid z(v,d)\,.\,(!\,\beta{\rightarrow}v \mid !\,d{\rightarrow}y) \mid & \\
\qquad\qquad !\,(\nu\alpha x)\,(!\,\llbracket R \rrbracket_{\mathrm{F}} \mid !\,\alpha{\rightarrow}x \mid !\,\llbracket P \rrbracket_{\mathrm{F}})) & \triangleq \\
\llbracket (Q\widehat{\alpha} \not\nearrow \widehat{x}P)\widehat{\beta}\,[z]\,\widehat{y}(R\widehat{\alpha} \not\nearrow \widehat{x}P) \rrbracket_{\mathrm{F}} &
\end{array}$$

($\not\backslash$*imp-outs*)**:** Then $P\widehat{\alpha} \not\backslash \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R) \rightarrow_{\mathcal{X}} P\widehat{\alpha} \dagger \widehat{z}((P\widehat{\alpha} \not\backslash \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \not\backslash \widehat{x}R))$.

$$\begin{array}{lll}
\llbracket P\widehat{\alpha} \not\backslash \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R)\rrbracket_{\mathrm{F}} & \triangleq & \\
(\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,(\nu\beta y)\,(!\,\llbracket Q\rrbracket_{\mathrm{F}} \mid x(v,d).\,(!\,\beta\rightarrow v \mid !\,d\rightarrow y) \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \sim_{\mathsf{c}} & (3.7\,(1b)) \\
!\,(\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid (\nu\beta y)\,(!\,\llbracket Q\rrbracket_{\mathrm{F}} \mid x(v,d).\,(!\,\beta\rightarrow v \mid !\,d\rightarrow y) \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \sim_{\mathsf{c}} & (3.7\,(3a)) \\
!\,(\nu\alpha z)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow z \mid (\nu\beta y)\,((\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,\llbracket Q\rrbracket_{\mathrm{F}}) \mid & & \\
\qquad z(v,d).\,(!\,\beta\rightarrow v \mid !\,d\rightarrow y) \mid (\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,\llbracket R\rrbracket_{\mathrm{F}}))) & \sim_{\mathsf{c}} & (3.7\,(1b)) \\
(\nu\alpha z)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow z \mid !\,(\nu\beta y)\,((\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,\llbracket Q\rrbracket_{\mathrm{F}}) \mid & & \\
\qquad z(v,d).\,(!\,\beta\rightarrow v \mid !\,d\rightarrow y) \mid (\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,\llbracket R\rrbracket_{\mathrm{F}}))) & \sim_{\mathsf{c}} & (3.7\,(1a)) \\
(\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow z \mid !\,(\nu\beta y)\,(!\,(\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,\llbracket Q\rrbracket_{\mathrm{F}}) \mid & & \\
\qquad z(v,d)\,.\,(!\,\beta\rightarrow v \mid !\,d\rightarrow y) \mid !\,(\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,\llbracket R\rrbracket_{\mathrm{F}}))) & \triangleq & \\
\llbracket P\widehat{\alpha} \dagger \widehat{z}((P\widehat{\alpha} \not\backslash \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \not\backslash \widehat{x}R))\rrbracket_{\mathrm{F}} & &
\end{array}$$

($\not\backslash$*imp-ins*)**:** Then $P\widehat{\alpha} \not\backslash \widehat{x}(Q\widehat{\beta}\,[z]\,\widehat{y}R) \rightarrow_{\mathcal{X}} (P\widehat{\alpha} \not\backslash \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \not\backslash \widehat{x}R), z \neq x$.

$$\begin{array}{lll}
\llbracket P\widehat{\alpha} \not\backslash \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R)\rrbracket_{\mathrm{F}} & \triangleq & \\
(\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,(\nu\beta y)\,(!\,\llbracket Q\rrbracket_{\mathrm{F}} \mid x(v,d).\,(!\,\beta\rightarrow v \mid !\,d\rightarrow y) \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \sim_{\mathsf{c}} & (3.7\,(1b)) \\
!\,(\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid (\nu\beta y)\,(!\,\llbracket Q\rrbracket_{\mathrm{F}} \mid z(v,d).\,(!\,\beta\rightarrow v \mid !\,d\rightarrow y) \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \sim_{\mathsf{c}} & (3.7\,(3a)) \\
!\,(\nu\beta y)\,((\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,\llbracket Q\rrbracket_{\mathrm{F}}) \mid & & \\
\qquad z(v,d).\,(!\,\beta\rightarrow v \mid !\,d\rightarrow y) \mid (\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & {}_{\mathsf{c}}\sqsupseteq & \\
(\nu\beta y)\,((\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,\llbracket Q\rrbracket_{\mathrm{F}}) \mid & & \\
\qquad z(v,d).\,(!\,\beta\rightarrow v \mid !\,d\rightarrow y) \mid (\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \sim_{\mathsf{c}} & (3.7\,(1a)) \\
(\nu\beta y)\,(!\,(\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,\llbracket Q\rrbracket_{\mathrm{F}}) \mid & & \\
\qquad z(v,d)\,.\,(!\,\beta\rightarrow v \mid !\,d\rightarrow y) \mid !\,(\nu\alpha x)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid !\,\alpha\rightarrow x \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \triangleq & \\
\llbracket (P\widehat{\alpha} \not\backslash \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \not\backslash \widehat{x}R)\rrbracket_{\mathrm{F}} & &
\end{array}$$

$P \rightarrow Q \Rightarrow P\widehat{\alpha}\,[x]\,\widehat{y}R \rightarrow Q\widehat{\alpha}\,[x]\,\widehat{y}R,\ R\widehat{\alpha}\,[x]\,\widehat{y}P \rightarrow R\widehat{\alpha}\,[x]\,\widehat{y}Q$**:** By induction. □

Example 5.4. Using this full encoding, we can now represent the last reduction of Example 2.8, *i.e.* that of

$$(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{u}(Q\widehat{\tau}\,[u]\,\widehat{w}R)$$

in π, where $P = \langle z\cdot\delta\rangle$, $Q = \langle u\cdot\tau\rangle$ and $R = \langle w\cdot\sigma\rangle$.

$$\begin{array}{lll}
\llbracket (\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{u}(Q\widehat{\tau}\,[u]\,\widehat{w}R)\rrbracket_{\mathrm{F}} & \triangleq & \\
(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma\rightarrow u \mid !\,\llbracket Q\widehat{\tau}\,[u]\,\widehat{w}R\rrbracket_{\mathrm{F}}) & \triangleq & \\
(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma\rightarrow u \mid !\,(\nu\tau w)\,(!\,Q \mid u(v,d).\,(!\,\tau\rightarrow v \mid !\,d\rightarrow w) \mid !\,R)) & \sim_{\mathsf{c}} & \mathit{(3.7)} \\
(\nu\gamma y)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma\rightarrow y \mid & & \\
\quad !\,(\nu\tau w)\,(!\,(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma\rightarrow u \mid !\,\llbracket Q\rrbracket_{\mathrm{F}}) \mid y(v,d)\,.\,(!\,\tau\rightarrow v \mid !\,d\rightarrow w) \mid & & \\
\qquad !\,(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma\rightarrow u \mid !\,\llbracket R\rrbracket_{\mathrm{F}}))) & {}_{\mathsf{c}}\sqsupseteq & (=_{\alpha}) \\
(\nu\gamma y)\,(!\,\llbracket \widehat{x}\langle x\cdot\rho\rangle\widehat{\rho}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma\rightarrow y \mid !\,(\nu\tau w)\,(!\,(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma\rightarrow u \mid & & \\
\qquad !\,u(w).\,\overline{\tau}\langle w\rangle) \mid y(v,d).\,(!\,\tau\rightarrow v \mid !\,d\rightarrow w) \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \rightarrow_{\pi} & (\gamma,u) \\
(\nu\gamma y)\,(!\,\llbracket \widehat{x}\langle x\cdot\rho\rangle\widehat{\rho}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma\rightarrow y \mid & & \\
\qquad !\,(\nu\tau w)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\tau\rrbracket_{\mathrm{F}} \mid y(v,d).\,(!\,\tau\rightarrow v \mid !\,d\rightarrow w) \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \triangleq & \\
\llbracket (\widehat{x}\langle x\cdot\rho\rangle\widehat{\rho}\cdot\gamma)\widehat{\gamma} \dagger \widehat{y}((\widehat{z}P\widehat{\delta}\cdot\tau)\widehat{\tau}\,[y]\,\widehat{w}R)\rrbracket_{\mathrm{F}} & {}_{\mathsf{c}}\sqsupseteq &
\end{array}$$

$$\begin{array}{lr}
(\nu\gamma y)\,((\nu x\rho)\,(!\,x(w).\overline{\rho}\langle w\rangle \mid \overline{\gamma}\langle x,\rho\rangle) \mid \gamma\rightarrow y \mid & \\
\qquad\qquad (\nu\tau w)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\tau\rrbracket_{\mathrm{F}} \mid y(v,d).\,(!\,\tau\rightarrow v \mid !\,d\rightarrow w) \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \rightarrow_\pi\ (\gamma,y) \\
(\nu\gamma y)\,((\nu x\rho)\,(!\,x(w).\overline{\rho}\langle w\rangle \mid (\nu\tau w)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\tau\rightarrow x \mid !\,\rho\rightarrow w \mid !\,\llbracket R\rrbracket_{\mathrm{F}}))) & \equiv \\
(\nu\tau x)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\tau\rrbracket_{\mathrm{F}} \mid !\,\tau\rightarrow x \mid !\,(\nu\rho w)\,(!\,x(w).\overline{\rho}\langle w\rangle \mid !\,\rho\rightarrow w \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \triangleq \\
\llbracket (\widehat{z}P\widehat{\delta}\cdot\tau)\widehat{\tau} \dagger \widehat{x}(\langle x\cdot\rho\rangle\widehat{\rho} \dagger \widehat{w}R)\rrbracket_{\mathrm{F}} & \sim_{\mathsf{c}}\ (3.7) \\
(\nu\rho w)\,(!\,(\nu\tau x)\,(!\,(\nu z\delta)\,(!\,P \mid \overline{\tau}\langle z,\delta\rangle) \mid !\,\tau\rightarrow x \mid !\,x(w).\overline{\rho}\langle w\rangle) \mid !\,\rho\rightarrow w \mid & \\
\qquad\qquad !\,(\nu\tau z)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\tau\rrbracket_{\mathrm{F}} \mid !\,\tau\rightarrow z \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & {}_{\mathsf{c}}\!\sqsupseteq \\
(\nu\rho w)\,(!\,(\nu\tau x)\,(!\,(\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid \overline{\tau}\langle z,\delta\rangle) \mid !\,\tau\rightarrow x \mid !\,x(w).\overline{\rho}\langle w\rangle) \mid !\,\rho\rightarrow w \mid !\,\llbracket R\rrbracket_{\mathrm{F}}) & \sim_{\mathsf{c}}\ (\tau,x) \\
(\nu\rho w)\,(!\,(\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid \overline{\rho}\langle z,\delta\rangle) \mid !\,\rho\rightarrow w \mid !\,\llbracket R\rrbracket_{\mathrm{F}}) & {}_{\mathsf{c}}\!\sqsupseteq \\
(\nu\rho w)\,((\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid \overline{\rho}\langle z,\delta\rangle) \mid \rho\rightarrow w \mid \llbracket R\rrbracket_{\mathrm{F}}) & \rightarrow_\pi\ (\rho,w) \\
(\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid \overline{\sigma}\langle z,\delta\rangle) &
\end{array}$$

We will now define the encoding $\llbracket\cdot\rrbracket_{\mathrm{R}}$ as a variant of $\llbracket\cdot\rrbracket_{\mathrm{F}}$; the main idea of this third encoding is to see terms as infinite resources rather than using replication to model substitution, so use inherent replication for all synchronisation. This is achieved by replicating all communication, *i.e.* all *input* and *output* actions. This replicated encoding is easier to understand, but differs from the other two in that it does not model reduction via reduction, but via contextual equality, whereas the other two truly use π's reduction in the proofs.

Definition 5.5 (Replicative encoding for $\mathcal{X}$ in π)**.**

$$\begin{array}{rcl}
\llbracket \langle x\cdot\alpha\rangle\rrbracket_{\mathrm{R}} & = & !\,x(w).\overline{\alpha}\langle w\rangle \\
\llbracket \widehat{y}Q\widehat{\beta}\cdot\alpha\rrbracket_{\mathrm{R}} & = & (\nu y\beta)\,(\llbracket Q\rrbracket_{\mathrm{R}} \mid !\,\overline{\alpha}\langle y,\beta\rangle) \\
\llbracket P\widehat{\alpha}\,[x]\,\widehat{y}Q\rrbracket_{\mathrm{R}} & = & (\nu\alpha y)\,(\llbracket P\rrbracket_{\mathrm{R}} \mid !\,x(s,d).\,(!\,\alpha\rightarrow s \mid !\,d\rightarrow y) \mid \llbracket Q\rrbracket_{\mathrm{R}}) \\
\llbracket P\widehat{\alpha} \dagger \widehat{x}Q\rrbracket_{\mathrm{R}} & = & (\nu\alpha x)\,(\llbracket P\rrbracket_{\mathrm{R}} \mid !\,\alpha\rightarrow x \mid \llbracket Q\rrbracket_{\mathrm{R}}) \qquad = \llbracket P\widehat{\alpha} \not\nearrow \widehat{x}Q\rrbracket_{\mathrm{R}} = \llbracket P\widehat{\alpha} \not\nwarrow \widehat{x}Q\rrbracket_{\mathrm{R}}
\end{array}$$

Notice that (as in Def. 4.1 and 5.1) all *cuts* are interpreted the same way.

This new approach will be as expressive as the full encoding we considered above, but has as advantage that it is more abstract and gives a better semantics in that the main proof follows more easily.

For this encoding, we can show that replication is implicit for encoded terms:

Lemma 5.6. $\llbracket P\rrbracket_{\mathrm{R}} \sim_{\mathsf{c}} !\,\llbracket P\rrbracket_{\mathrm{R}}$.

Proof. By induction on the structure of terms.

$$\begin{array}{ll}
P = \langle x\cdot\alpha\rangle\textbf{:} & \llbracket \langle x\cdot\alpha\rangle\rrbracket_{\mathrm{R}} \triangleq !\,x(w).\overline{\alpha}\langle w\rangle \equiv !\,!\,x(w).\overline{\alpha}\langle w\rangle \triangleq !\,\llbracket \langle x\cdot\alpha\rangle\rrbracket_{\mathrm{R}}. \\
P = \widehat{x}Q\widehat{\alpha}\cdot\beta\textbf{:} & \llbracket \widehat{x}Q\widehat{\alpha}\cdot\beta\rrbracket_{\mathrm{R}} \triangleq (\nu x\alpha)\,(\llbracket Q\rrbracket_{\mathrm{R}} \mid !\,\overline{\beta}\langle x,\alpha\rangle) \equiv (\mathit{IH})\ (\nu x\alpha)\ (!\,\llbracket Q\rrbracket_{\mathrm{R}} \mid !\,!\,\overline{\beta}\langle x,\alpha\rangle) \\
& \quad \sim_{\mathsf{c}} (3.7\,(1\mathrm{a}))\ !\,(\nu x\alpha)\,(\llbracket Q\rrbracket_{\mathrm{R}} \mid !\,\overline{\beta}\langle x,\alpha\rangle) \triangleq !\,\llbracket \widehat{x}Q\widehat{\alpha}\cdot\beta\rrbracket_{\mathrm{R}} \\
P = Q\widehat{\alpha}\,[y]\,\widehat{x}R\textbf{:} & \llbracket Q\widehat{\alpha}\,[y]\,\widehat{x}R\rrbracket_{\mathrm{R}} \triangleq (\nu\alpha x)\,(\llbracket Q\rrbracket_{\mathrm{R}} \mid !\,y(s,d).\,(!\,\alpha\rightarrow s \mid !\,d\rightarrow x) \mid \llbracket R\rrbracket_{\mathrm{R}}) \equiv (\mathit{IH}) \\
& (\nu x\alpha)\ (!\,\llbracket Q\rrbracket_{\mathrm{R}} \mid !\,!\,y(v,d).\,(!\,\alpha\rightarrow v \mid !\,d\rightarrow x)!\,\llbracket R\rrbracket_{\mathrm{R}}) \sim_{\mathsf{c}} (3.7\,(1\mathrm{a})) \\
& !\,(\nu\alpha x)\,(\llbracket Q\rrbracket_{\mathrm{R}} \mid !\,y(s,d).\,(!\,\alpha\rightarrow s \mid !\,d\rightarrow x) \mid \llbracket R\rrbracket_{\mathrm{R}}) \triangleq !\,\llbracket Q\widehat{\alpha}\,[y]\,\widehat{x}R\rrbracket_{\mathrm{R}} \\
P = Q\widehat{\alpha} \dagger \widehat{x}R\textbf{:} & \llbracket Q\widehat{\alpha} \dagger \widehat{x}R\rrbracket_{\mathrm{R}} \triangleq (\nu\alpha x)\,(\llbracket Q\rrbracket_{\mathrm{R}} \mid !\,\alpha\rightarrow x \mid \llbracket R\rrbracket_{\mathrm{R}}) \equiv (\mathit{IH})\ (\nu x\alpha)\ (!\,\llbracket Q\rrbracket_{\mathrm{R}} \mid !\,!\,\alpha\rightarrow x!\,\llbracket R\rrbracket_{\mathrm{R}}) \\
& \quad \sim_{\mathsf{c}} (3.7\,(1\mathrm{a}))\ !\,(\nu\alpha x)\,(\llbracket Q\rrbracket_{\mathrm{R}} \mid !\,\alpha\rightarrow x \mid \llbracket R\rrbracket_{\mathrm{R}}) \triangleq !\,\llbracket Q\widehat{\alpha} \dagger \widehat{x}R\rrbracket_{\mathrm{R}}
\end{array}$$

Since by this lemma replication is implicitly used everywhere, we no longer relate two terms via reduction: it is clear that $(\nu a)\,(!\,a(x).\,P \mid !\,\overline{a}\langle b\rangle)$ is equivalent to $!\,P[b/x]$, but via a reduction we can at most show:

$$\begin{array}{lll}
(\nu a)\,(!\,a(x).P \mid !\,\overline{a}\langle b\rangle) & \equiv & \\
(\nu a)\,(!\,a(x).P \mid !\,\overline{a}\langle b\rangle \mid a(x).P \mid \overline{a}\langle b\rangle) & \rightarrow_\pi & \\
(\nu a)\,(!\,a(x).P \mid !\,\overline{a}\langle b\rangle) \mid P[b/x] & \not= & !\,P[b/x]
\end{array}$$

We use this lemma in the next result, when we apply Lemma 3.7's distribution rules.

Theorem 5.7. *If $P \rightarrow_{\mathcal{X}} Q$, then $\llbracket P \rrbracket_{\mathrm{R}} \;{}_{\mathsf{C}}\!\sqsupseteq\; \llbracket Q \rrbracket_{\mathrm{R}}$.*

Proof. **Logical rules:**

(cap)**:** $\langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle \rightarrow \langle y{\cdot}\gamma\rangle$.

$$\begin{array}{lll}
\llbracket \langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle \rrbracket_{\mathrm{R}} \triangleq & (\nu\alpha x)\,(\llbracket \langle y{\cdot}\alpha\rangle \rrbracket_{\mathrm{R}} \mid !\,\alpha\rightarrow x \mid \llbracket \langle x{\cdot}\gamma\rangle \rrbracket_{\mathrm{R}}) & \triangleq \\
& (\nu\alpha x)\,(!\,y(w).\overline{\alpha}\langle w\rangle \mid !\,\alpha\rightarrow x \mid !\,x(w).\overline{\gamma}\langle w\rangle) & \equiv \\
& !\,y(w).\overline{\alpha}\langle w\rangle \mid (\nu\alpha x)\,(!\,\alpha\rightarrow x \mid !\,x(w).\overline{\gamma}\langle w\rangle) & \sim_{\mathsf{C}} \\
& !\,y(w).\overline{\gamma}\langle w\rangle & \triangleq \; \llbracket \langle y{\cdot}\gamma\rangle \rrbracket_{\mathrm{R}}
\end{array}$$

(exp)**:** $(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle \rightarrow \widehat{y}P\widehat{\beta}{\cdot}\gamma$.

$$\begin{array}{ll}
\llbracket (\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle \rrbracket_{\mathrm{R}} \triangleq (\nu\alpha x)\,(\llbracket \widehat{y}P\widehat{\beta}{\cdot}\alpha \rrbracket_{\mathrm{R}} \mid !\,\alpha\rightarrow x \mid \llbracket \langle x{\cdot}\gamma\rangle \rrbracket_{\mathrm{R}}) & \triangleq \\
(\nu\alpha x)\,((\nu y\beta)\,(\llbracket P \rrbracket_{\mathrm{R}} \mid !\,\overline{\alpha}\langle y,\beta\rangle) \mid !\,\alpha\rightarrow x \mid !\,x(w).\overline{\gamma}\langle w\rangle) & \sim_{\mathsf{C}} \; (\alpha,x) \\
(\nu y\beta)\,(\llbracket P \rrbracket_{\mathrm{R}} \mid !\,\overline{\gamma}\langle y,\beta\rangle) & \triangleq \; \llbracket \widehat{y}P\widehat{\beta}{\cdot}\gamma \rrbracket_{\mathrm{R}}
\end{array}$$

(imp)**:** $\langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{z}P) \rightarrow Q\widehat{\beta}\,[y]\,\widehat{z}P$.

$$\begin{array}{ll}
\llbracket \langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{z}P) \rrbracket_{\mathrm{R}} \triangleq (\nu\alpha x)\,(\llbracket \langle y{\cdot}\alpha\rangle \rrbracket_{\mathrm{R}} \mid !\,\alpha\rightarrow x \mid \llbracket Q\widehat{\beta}\,[x]\,\widehat{z}P \rrbracket_{\mathrm{R}}) & \triangleq \\
(\nu\alpha x)\,(!\,y(w).\overline{\alpha}\langle w\rangle \mid !\,\alpha\rightarrow x \mid (\nu\beta z)\,(\llbracket Q \rrbracket_{\mathrm{R}} \mid !\,x(s,d).(!\,\beta\rightarrow s \mid !\,d\rightarrow z) \mid \llbracket P \rrbracket_{\mathrm{R}})) & \sim_{\mathsf{C}} \; (\alpha,x) \\
(\nu\beta z)\,(\llbracket Q \rrbracket_{\mathrm{R}} \mid !\,y(s,d).(!\,\beta\rightarrow s \mid !\,d\rightarrow z) \mid \llbracket P \rrbracket_{\mathrm{R}}) & \triangleq \; \llbracket Q\widehat{\beta}\,[y]\,\widehat{z}P \rrbracket_{\mathrm{R}}
\end{array}$$

$(exp\text{-}imp)$**:** $(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \rightarrow Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R)$.

$$\begin{array}{ll}
\llbracket (\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \rrbracket_{\mathrm{R}} \triangleq (\nu\alpha x)\,(\llbracket \widehat{y}P\widehat{\beta}{\cdot}\alpha \rrbracket_{\mathrm{R}} \mid !\,\alpha\rightarrow x \mid \llbracket Q\widehat{\gamma}\,[x]\,\widehat{z}R \rrbracket_{\mathrm{R}}) & \triangleq \\
(\nu\alpha x)\,((\nu y\beta)\,(\llbracket P \rrbracket_{\mathrm{R}} \mid !\,\overline{\alpha}\langle y,\beta\rangle) \mid !\,\alpha\rightarrow x \mid & \\
\qquad\qquad (\nu\gamma z)\,(\llbracket Q \rrbracket_{\mathrm{R}} \mid !\,x(s,d).(!\,\gamma\rightarrow s \mid !\,d\rightarrow z) \mid \llbracket R \rrbracket_{\mathrm{R}})) & \sim_{\mathsf{C}} \; (\alpha,x) \\
(\nu y\beta\gamma z)\,(\llbracket P \rrbracket_{\mathrm{R}} \mid \llbracket Q \rrbracket_{\mathrm{R}} \mid !\,\gamma\rightarrow y \mid !\,\beta\rightarrow z \mid \llbracket R \rrbracket_{\mathrm{R}}) & \equiv \\
(\nu\gamma y)\,(\llbracket Q \rrbracket_{\mathrm{R}} \mid !\,\gamma\rightarrow y \mid (\nu\beta z)\,(\llbracket P \rrbracket_{\mathrm{R}} \mid !\,\beta\rightarrow z \mid \llbracket R \rrbracket_{\mathrm{R}})) & \triangleq \\
(\nu\gamma y)\,(\llbracket Q \rrbracket_{\mathrm{R}} \mid !\,\gamma\rightarrow y \mid \llbracket P\widehat{\beta} \dagger \widehat{z}R \rrbracket_{\mathrm{R}}) & \triangleq \llbracket Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R) \rrbracket_{\mathrm{R}}
\end{array}$$

For $(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \rightarrow (Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R$ the proof is similar:

$$\begin{array}{ll}
(\nu y\beta\gamma z)\,(\llbracket P \rrbracket_{\mathrm{R}} \mid \llbracket Q \rrbracket_{\mathrm{R}} \mid !\,\gamma\rightarrow y \mid !\,\beta\rightarrow z \mid \llbracket R \rrbracket_{\mathrm{R}}) & \equiv \\
(\nu\beta z)\,((\nu\gamma y)\,(\llbracket Q \rrbracket_{\mathrm{R}} \mid !\,\gamma\rightarrow y \mid \llbracket P \rrbracket_{\mathrm{R}}) \mid !\,\beta\rightarrow z \mid \llbracket R \rrbracket_{\mathrm{R}}) & \triangleq \\
(\nu\beta z)\,(\llbracket Q\widehat{\gamma} \dagger \widehat{y}P \rrbracket_{\mathrm{R}} \mid !\,\beta\rightarrow z \mid \llbracket R \rrbracket_{\mathrm{R}}) & \triangleq \; \llbracket (Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R \rrbracket_{\mathrm{R}}
\end{array}$$

Activation rules:

$(a\not\dagger)$**:** $P\widehat{\alpha} \dagger \widehat{x}Q \rightarrow P\widehat{\alpha} \not\dagger \widehat{x}Q$, if P does not introduce α. Since both are interpreted via $(\nu\alpha x)\,(\llbracket P \rrbracket_{\mathrm{R}} \mid !\,\alpha\rightarrow x \mid \llbracket Q \rrbracket_{\mathrm{R}})$, this is immediate.

$(\backslash\!\!\!\dagger a)$**:** $P\widehat{\alpha} \dagger \widehat{x}Q \rightarrow P\widehat{\alpha} \,\backslash\!\!\!\dagger\, \widehat{x}Q$, if Q does not introduce x. Similar.

Left propagation:

$(d\not\dagger)$**:** $\langle y{\cdot}\alpha\rangle\widehat{\alpha} \not\dagger \widehat{x}P \rightarrow \langle y{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}P$. Similar.

$(\mathit{cap}\not\nearrow)$: $\langle y{\cdot}\beta\rangle\widehat{\alpha}\not\nearrow\widehat{x}P \rightarrow \langle y{\cdot}\beta\rangle$, $\beta \neq \alpha$.

$$\begin{aligned}
[\![\langle y{\cdot}\beta\rangle\widehat{\alpha}\not\nearrow\widehat{x}P]\!]_{\mathrm{R}} &\triangleq (\nu\alpha x)\,([\![\langle y{\cdot}\beta\rangle]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid [\![P]\!]_{\mathrm{R}}) \;\equiv\; (\beta\neq\alpha)\\
&\ [\![\langle y{\cdot}\beta\rangle]\!]_{\mathrm{R}} \mid (\nu\alpha x)\,(!\,\alpha{\rightarrow}x \mid [\![P]\!]_{\mathrm{R}}) \;{}_{\mathsf{c}}\!\sqsupseteq\; [\![\langle y{\cdot}\beta\rangle]\!]_{\mathrm{R}}
\end{aligned}$$

$(\mathit{exp\text{-}outs}\not\nearrow)$: $(\widehat{y}Q\widehat{\beta}{\cdot}\alpha)\widehat{\alpha}\not\nearrow\widehat{x}P \rightarrow (\widehat{y}(Q\widehat{\alpha}\not\nearrow\widehat{x}P)\widehat{\beta}{\cdot}\gamma)\widehat{\gamma}\dagger\widehat{x}P$, γ *fresh*.

$$\begin{aligned}
&[\![(\widehat{y}Q\widehat{\beta}{\cdot}\alpha)\widehat{\alpha}\not\nearrow\widehat{x}P]\!]_{\mathrm{R}} \triangleq (\nu\alpha x)\,((\nu y\beta)\,([\![Q]\!]_{\mathrm{R}} \mid !\,\overline{\alpha}\langle y,\beta\rangle) \mid !\,\alpha{\rightarrow}x \mid [\![P]\!]_{\mathrm{R}}) \;\sim_{\mathsf{c}}\; (3.7\,(2b))\\
&(\nu\gamma x)\,((\nu y\beta)\,((\nu\alpha x)\,([\![Q]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid [\![P]\!]_{\mathrm{R}}) \mid !\,\overline{\gamma}\langle y,\beta\rangle) \mid !\,\gamma{\rightarrow}x \mid [\![P]\!]_{\mathrm{R}}) \;\triangleq\\
&[\![(\widehat{y}(Q\widehat{\alpha}\not\nearrow\widehat{x}P)\widehat{\beta}{\cdot}\gamma)\widehat{\gamma}\dagger\widehat{x}P]\!]_{\mathrm{R}}
\end{aligned}$$

$(\mathit{exp\text{-}ins}\not\nearrow)$: $(\widehat{y}Q\widehat{\beta}{\cdot}\gamma)\widehat{\alpha}\not\nearrow\widehat{x}P \rightarrow \widehat{y}(Q\widehat{\alpha}\not\nearrow\widehat{x}P)\widehat{\beta}{\cdot}\gamma$, $\gamma\neq\alpha$.

$$\begin{aligned}
[\![(\widehat{y}Q\widehat{\beta}{\cdot}\gamma)\widehat{\alpha}\not\nearrow\widehat{x}P]\!]_{\mathrm{R}} &\triangleq (\nu\alpha x)\,((\nu y\beta)\,([\![Q]\!]_{\mathrm{R}} \mid !\,\overline{\gamma}\langle y,\beta\rangle) \mid !\,\alpha{\rightarrow}x \mid [\![P]\!]_{\mathrm{R}}) \;\equiv\; (\gamma\neq\alpha)\\
&\ (\nu y\beta)\,((\nu\alpha x)\,([\![Q]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid [\![P]\!]_{\mathrm{R}}) \mid !\,\overline{\gamma}\langle y,\beta\rangle) \;\triangleq\; [\![\widehat{y}(Q\widehat{\alpha}\not\nearrow\widehat{x}P)\widehat{\beta}{\cdot}\gamma]\!]_{\mathrm{R}}
\end{aligned}$$

$(\mathit{imp}\not\nearrow)$: $(Q\widehat{\beta}\,[z]\,\widehat{y}R)\widehat{\alpha}\not\nearrow\widehat{x}P \rightarrow (Q\widehat{\alpha}\not\nearrow\widehat{x}P)\widehat{\beta}\,[z]\,\widehat{y}(R\widehat{\alpha}\not\nearrow\widehat{x}P)$.

$$\begin{aligned}
&[\![(Q\widehat{\beta}\,[z]\,\widehat{y}R)\widehat{\alpha}\not\nearrow\widehat{x}P]\!]_{\mathrm{R}} && \triangleq\\
&(\nu\alpha x)\,((\nu\beta y)\,([\![Q]\!]_{\mathrm{R}} \mid !\,z(s,d).\,(!\,\beta{\rightarrow}s \mid !\,d{\rightarrow}y) \mid [\![R]\!]_{\mathrm{R}}) \mid !\,\alpha{\rightarrow}x \mid [\![P]\!]_{\mathrm{R}}) && \sim_{\mathsf{c}}\;(3.7\,(2a))\\
&(\nu\beta y)\,((\nu\alpha x)\,([\![Q]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid [\![P]\!]_{\mathrm{R}}) \mid\\
&\qquad\qquad !\,z(v,d).\,(!\,\beta{\rightarrow}v \mid !\,d{\rightarrow}y) \mid (\nu\alpha x)\,([\![R]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid [\![P]\!]_{\mathrm{R}})) && \triangleq\\
&(\nu\beta y)\,([\![Q\widehat{\alpha}\dagger\widehat{x}P]\!]_{\mathrm{R}} \mid !\,z(s,d).\,(!\,\beta{\rightarrow}s \mid !\,d{\rightarrow}y) \mid [\![R\widehat{\alpha}\dagger\widehat{x}P]\!]_{\mathrm{R}}) && \triangleq\\
&[\![(Q\widehat{\alpha}\not\nearrow\widehat{x}P)\widehat{\beta}\,[z]\,\widehat{y}(R\widehat{\alpha}\not\nearrow\widehat{x}P)]\!]_{\mathrm{R}}
\end{aligned}$$

$(\mathit{cut}\not\nearrow)$: $(Q\widehat{\beta}\dagger\widehat{y}R)\widehat{\alpha}\not\nearrow\widehat{x}P \rightarrow (Q\widehat{\alpha}\not\nearrow\widehat{x}P)\widehat{\beta}\dagger\widehat{y}(R\widehat{\alpha}\not\nearrow\widehat{x}P)$.

$$\begin{aligned}
&[\![(Q\widehat{\beta}\dagger\widehat{y}R)\widehat{\alpha}\not\nearrow\widehat{x}P]\!]_{\mathrm{R}} \triangleq (\nu\alpha x)\,((\nu\beta y)\,([\![Q]\!]_{\mathrm{R}} \mid !\,\beta{\rightarrow}y \mid [\![R]\!]_{\mathrm{R}}) \mid !\,\alpha{\rightarrow}x \mid [\![P]\!]_{\mathrm{R}}) \;\sim_{\mathsf{c}}\; (3.7\,(2a))\\
&(\nu\beta y)\,((\nu\alpha x)\,([\![Q]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid [\![R]\!]_{\mathrm{R}}) \mid !\,\beta{\rightarrow}y \mid (\nu\alpha x)\,([\![R]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid [\![P]\!]_{\mathrm{R}})) \qquad \triangleq\\
&[\![(Q\widehat{\alpha}\not\nearrow\widehat{x}P)\widehat{\beta}\dagger\widehat{y}(R\widehat{\alpha}\not\nearrow\widehat{x}P)]\!]_{\mathrm{R}}
\end{aligned}$$

Right propagation:

$(\not\nwarrow d)$: $P\widehat{\alpha}\not\nwarrow\widehat{x}\langle x{\cdot}\beta\rangle \rightarrow P\widehat{\alpha}\dagger\widehat{x}\langle x{\cdot}\beta\rangle$, α *not introduced in* P. As above.

$(\not\nwarrow\mathit{cap})$: $P\widehat{\alpha}\not\nwarrow\widehat{x}\langle y{\cdot}\beta\rangle \rightarrow \langle y{\cdot}\beta\rangle$, $y\neq x$.

$$\begin{aligned}
[\![P\widehat{\alpha}\not\nwarrow\widehat{x}\langle y{\cdot}\beta\rangle]\!]_{\mathrm{R}} &\triangleq (\nu\alpha x)\,([\![P]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid [\![\langle y{\cdot}\beta\rangle]\!]_{\mathrm{R}}) \;\equiv\; (y\neq x)\\
&\ (\nu\alpha x)\,([\![P]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x) \mid [\![\langle y{\cdot}\beta\rangle]\!]_{\mathrm{R}} \;{}_{\mathsf{c}}\!\sqsupseteq\; [\![\langle y{\cdot}\beta\rangle]\!]_{\mathrm{R}}
\end{aligned}$$

$(\not\nwarrow\mathit{exp})$: $P\widehat{\alpha}\not\nwarrow\widehat{x}(\widehat{y}Q\widehat{\beta}{\cdot}\gamma) \rightarrow \widehat{y}(P\widehat{\alpha}\not\nwarrow\widehat{x}Q)\widehat{\beta}{\cdot}\gamma$.

$$\begin{aligned}
[\![P\widehat{\alpha}\not\nwarrow\widehat{x}(\widehat{y}Q\widehat{\beta}{\cdot}\gamma)]\!]_{\mathrm{R}} &\triangleq (\nu\alpha x)\,([\![P]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid (\nu y\beta)\,([\![Q]\!]_{\mathrm{R}} \mid !\,\overline{\gamma}\langle y,\beta\rangle)) \;\equiv\\
&\ (\nu y\beta)\,((\nu\alpha x)\,([\![P]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid [\![Q]\!]_{\mathrm{R}}) \mid !\,\overline{\gamma}\langle y,\beta\rangle) \;\triangleq\\
&\ (\nu y\beta)\,([\![P\widehat{\alpha}\not\nwarrow\widehat{x}Q]\!]_{\mathrm{R}} \mid !\,\overline{\gamma}\langle y,\beta\rangle) \qquad \triangleq\; [\![\widehat{y}(P\widehat{\alpha}\not\nwarrow\widehat{x}Q)\widehat{\beta}{\cdot}\gamma]\!]_{\mathrm{R}}
\end{aligned}$$

$(\not\nwarrow\mathit{imp\text{-}outs})$: $P\widehat{\alpha}\not\nwarrow\widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R) \rightarrow_{\mathcal{X}} P\widehat{\alpha}\dagger\widehat{z}((P\widehat{\alpha}\not\nwarrow\widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha}\not\nwarrow\widehat{x}R))$.

$$\begin{aligned}
&[\![P\widehat{\alpha}\not\nwarrow\widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R)]\!]_{\mathrm{R}} && \triangleq\\
&(\nu\alpha x)\,([\![P]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid (\nu\beta y)\,([\![Q]\!]_{\mathrm{R}} \mid !\,x(s,d).\,(!\,\beta{\rightarrow}s \mid !\,d{\rightarrow}y) \mid [\![R]\!]_{\mathrm{R}})) && \sim_{\mathsf{c}}\;(3.7\,(3a)\&(3b))\\
&(\nu\alpha z)\,(!\,[\![P]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}z \mid (\nu\beta y)\,((\nu\alpha x)\,([\![P]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid [\![Q]\!]_{\mathrm{R}}) \mid\\
&\qquad\qquad !\,z(v,d).\,(!\,\beta{\rightarrow}v \mid !\,d{\rightarrow}y) \mid (\nu\alpha x)\,([\![P]\!]_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid [\![R]\!]_{\mathrm{R}})) && \triangleq\\
&[\![P\widehat{\alpha}\dagger\widehat{z}((P\widehat{\alpha}\not\nwarrow\widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha}\not\nwarrow\widehat{x}R))]\!]_{\mathrm{R}}
\end{aligned}$$

$(\nwarrow imp\text{-}ins)$: $P\widehat{\alpha} \nwarrow \widehat{x}(Q\widehat{\beta}\,[z]\,\widehat{y}R) \to_{\mathcal{X}} (P\widehat{\alpha} \nwarrow \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \nwarrow \widehat{x}R),\ z \neq x.$

$$\begin{array}{ll}
\llbracket P\widehat{\alpha} \nwarrow \widehat{x}(Q\widehat{\beta}\,[z]\,\widehat{y}R)\rrbracket_{\mathrm{R}} & \triangleq \\
(\nu\alpha x)\,(\llbracket P\rrbracket_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid (\nu\beta y)\,(\llbracket Q\rrbracket_{\mathrm{R}} \mid !\,z(s,d).\,(!\,\beta{\rightarrow}s \mid !\,d{\rightarrow}y) \mid \llbracket R\rrbracket_{\mathrm{R}})) & \sim_{\mathsf{C}}\ (3.7\,(3a)) \\
(\nu\beta y)\,((\nu\alpha x)\,(\llbracket P\rrbracket_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid \llbracket Q\rrbracket_{\mathrm{R}}) \mid !\,z(s,d).\,(!\,\beta{\rightarrow}s \mid !\,d{\rightarrow}y) \mid (\nu\alpha x)\,(\llbracket P\rrbracket_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid \llbracket R\rrbracket_{\mathrm{R}})) & \triangleq \\
\llbracket (P\widehat{\alpha} \nwarrow \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \nwarrow \widehat{x}R)\rrbracket_{\mathrm{R}} &
\end{array}$$

$(\nwarrow cut)$: $P\widehat{\alpha} \nwarrow \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R) \to_{\mathcal{X}} (P\widehat{\alpha} \nwarrow \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \nwarrow \widehat{x}R).$

$$\begin{array}{ll}
\llbracket P\widehat{\alpha} \nwarrow \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R)\rrbracket_{\mathrm{R}} \triangleq (\nu\alpha x)\,(\llbracket P\rrbracket_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid (\nu\beta y)\,(\llbracket Q\rrbracket_{\mathrm{R}} \mid !\,\beta{\rightarrow}y \mid \llbracket R\rrbracket_{\mathrm{R}})) & \sim_{\mathsf{C}}\ (3.7\,(3a)) \\
(\nu\beta y)\,((\nu\alpha x)\,(\llbracket P\rrbracket_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid \llbracket Q\rrbracket_{\mathrm{R}}) \mid !\,\beta{\rightarrow}y \mid (\nu\alpha x)\,(\llbracket P\rrbracket_{\mathrm{R}} \mid !\,\alpha{\rightarrow}x \mid \llbracket R\rrbracket_{\mathrm{R}})) & \triangleq \\
\llbracket (P\widehat{\alpha} \nwarrow \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \nwarrow \widehat{x}R)\rrbracket_{\mathrm{R}} &
\end{array}$$

The contextual rules follow by induction. ☐

Notice that part (1) of Lemma 3.7 is not needed in this proof, and that ${}_{\mathsf{C}}\!\sqsupseteq$ is only needed in part $(cap \not\nearrow)$ and $(\nwarrow cap)$.

This concludes our simulation results. We have shown that our simple interpretation respects $\mathcal{X}$'s head-reduction, albeit via a contextual equivalence and perhaps leaving some additional processes running in parallel, and that full $\mathcal{X}$-reduction is respected by the full and replicative encodings.

Example 5.8. Simulating the third reduction of Example 2.8 using the full interpretation runs as follows:

$$\begin{array}{ll}
\llbracket (\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{u}(Q\widehat{\tau}\,[u]\,\widehat{w}R)\rrbracket_{\mathrm{F}} & \triangleq \\
(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma{\rightarrow}u \mid !\,\llbracket Q\widehat{\tau}\,[u]\,\widehat{w}R\rrbracket_{\mathrm{F}}) & \triangleq \\
(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma{\rightarrow}u \mid !\,(\nu\tau w)\,(!\,Q \mid u(v,d).\,(!\,\tau{\rightarrow}v \mid !\,d{\rightarrow}w) \mid !\,R)) & \sim_{\mathsf{C}}\ (3.7) \\
(\nu\gamma y)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma{\rightarrow}y \mid & \\
\qquad !\,(\nu\tau w)\,(!\,(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma{\rightarrow}u \mid !\,\llbracket Q\rrbracket_{\mathrm{F}}) \mid y(v,d)\,.\,(!\,\tau{\rightarrow}v \mid !\,d{\rightarrow}w) \mid & \\
\qquad\qquad !\,(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma{\rightarrow}u \mid !\,\llbracket R\rrbracket_{\mathrm{F}}))) & {}_{\mathsf{C}}\!\sqsupseteq\ (=_{\alpha}) \\
(\nu\gamma y)\,(!\,\llbracket \widehat{x}\langle x\cdot\rho\rangle\widehat{\rho}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma{\rightarrow}y \mid !\,(\nu\tau w)\,(!\,(\nu\gamma u)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma{\rightarrow}u \mid & \\
\qquad\qquad !\,u(w).\,\overline{\tau}\langle w\rangle) \mid y(v,d).\,(!\,\tau{\rightarrow}v \mid !\,d{\rightarrow}w) \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \to_{\pi}\ (\gamma,u) \\
(\nu\gamma y)\,(!\,\llbracket \widehat{x}\langle x\cdot\rho\rangle\widehat{\rho}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\gamma{\rightarrow}y \mid !\,(\nu\tau w)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\tau\rrbracket_{\mathrm{F}} \mid y(v,d).\,(!\,\tau{\rightarrow}v \mid !\,d{\rightarrow}w) \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \triangleq \\
\llbracket (\widehat{x}\langle x\cdot\rho\rangle\widehat{\rho}\cdot\gamma)\widehat{\gamma} \dagger \widehat{y}((\widehat{z}P\widehat{\delta}\cdot\tau)\widehat{\tau}\,[y]\,\widehat{w}R)\rrbracket_{\mathrm{F}} & {}_{\mathsf{C}}\!\sqsupseteq \\
(\nu\gamma y)\,((\nu x\rho)\,(!\,x(w).\,\overline{\rho}\langle w\rangle \mid \overline{\gamma}\langle x,\rho\rangle) \mid \gamma{\rightarrow}y \mid & \\
\qquad\qquad (\nu\tau w)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\tau\rrbracket_{\mathrm{F}} \mid y(v,d).\,(!\,\tau{\rightarrow}v \mid !\,d{\rightarrow}w) \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \to_{\pi}\ (\gamma,y) \\
(\nu\gamma y)\,((\nu x\rho)\,(!\,x(w).\,\overline{\rho}\langle w\rangle \mid (\nu\tau w)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{F}} \mid !\,\tau{\rightarrow}x \mid !\,\rho{\rightarrow}w \mid !\,\llbracket R\rrbracket_{\mathrm{F}}))) & \equiv \\
(\nu\tau x)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\tau\rrbracket_{\mathrm{F}} \mid !\,\tau{\rightarrow}x \mid !\,(\nu\rho w)\,(!\,x(w).\,\overline{\rho}\langle w\rangle \mid !\,\rho{\rightarrow}w \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & \triangleq \\
\llbracket (\widehat{z}P\widehat{\delta}\cdot\tau)\widehat{\tau} \dagger \widehat{x}(\langle x\cdot\rho\rangle\widehat{\rho} \dagger \widehat{w}R)\rrbracket_{\mathrm{F}} & \sim_{\mathsf{C}}\ (3.7) \\
(\nu\rho w)\,(!\,(\nu\tau x)\,(!\,(\nu z\delta)\,(!\,P \mid \overline{\tau}\langle z,\delta\rangle) \mid !\,\tau{\rightarrow}x \mid !\,x(w).\,\overline{\rho}\langle w\rangle) \mid !\,\rho{\rightarrow}w \mid & \\
\qquad\qquad !\,(\nu\tau z)\,(!\,\llbracket \widehat{z}P\widehat{\delta}\cdot\tau\rrbracket_{\mathrm{F}} \mid !\,\tau{\rightarrow}z \mid !\,\llbracket R\rrbracket_{\mathrm{F}})) & {}_{\mathsf{C}}\!\sqsupseteq \\
(\nu\rho w)\,(!\,(\nu\tau x)\,(!\,(\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid \overline{\tau}\langle z,\delta\rangle) \mid !\,\tau{\rightarrow}x \mid !\,x(w).\,\overline{\rho}\langle w\rangle) \mid !\,\rho{\rightarrow}w \mid !\,\llbracket R\rrbracket_{\mathrm{F}}) & \sim_{\mathsf{C}}\ (\tau,x) \\
(\nu\rho w)\,(!\,(\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid \overline{\rho}\langle z,\delta\rangle) \mid !\,\rho{\rightarrow}w \mid !\,\llbracket R\rrbracket_{\mathrm{F}}) & {}_{\mathsf{C}}\!\sqsupseteq \\
(\nu\rho w)\,((\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid \overline{\rho}\langle z,\delta\rangle) \mid \rho{\rightarrow}w \mid \llbracket R\rrbracket_{\mathrm{F}}) & \to_{\pi}(\rho,w) \\
(\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{F}} \mid \overline{\sigma}\langle z,\delta\rangle) &
\end{array}$$

6. Type assignment

In this section, we introduce a notion of type assignment for processes in π as presented in [8] that describes the '*input-output interface*' of a process. We will show that, if P is a witness to a judgement (in $\vdash_{\mathcal{X}}$), then its interpretations via $[\![\cdot]\!]_{\mathrm{S}}$, $[\![\cdot]\!]_{\mathrm{F}}$ and $[\![\cdot]\!]_{\mathrm{R}}$ are as well (in $\vdash_\pi$). Together with the preservation results we have shown above, this implies that we can encode proofs in LK to typeable processes, and have modelled *cut*-elimination - which transforms a proof into a proof. For the simple encoding, the notion of *cut*-elimination that has been modelled is that of head-reduction, but for the other two encodings, *cut*-elimination has been modelled in full.

Our notion is different in that it assigns to channels the type of the input or output that is sent over the channel; in that it differs from normal notions, that would state:

$$\frac{}{\overline{a}\langle b\rangle : \Gamma, b{:}A \vdash a{:}\mathsf{ch}(A), \Delta} \qquad \text{or} \qquad \frac{}{\overline{a}\langle b\rangle : \Gamma, b{:}A \vdash a{:}[A], \Delta}$$

In order to be able to encode LK, types in our system need not be decorated with channel information.

Definition 6.1 (Type assignment [8])**.** The types and contexts we consider for the π-calculus are defined like those of Definition 1.3, generalised to names, but allowing both Roman and Greek names on *both* sides.

Type assignment for π-calculus is defined by the following sequent system:

$$(0) : \frac{}{0 : \Gamma \vdash_\pi \Delta} \qquad (\nu) : \frac{P : \Gamma, a{:}A \vdash_\pi a{:}A, \Delta}{(\nu a)\, P : \Gamma \vdash_\pi \Delta}$$

$$(out) : \frac{}{\overline{a}\langle b\rangle : \Gamma, b{:}A \vdash_\pi a{:}A, b{:}A, \Delta}\ (b \neq a) \qquad (in) : \frac{P : \Gamma, x{:}A \vdash_\pi x{:}A, \Delta}{a(x).\, P : \Gamma, a{:}A \vdash_\pi \Delta}$$

$$(\mid) : \frac{P_i : \Gamma \vdash_\pi \Delta \quad (i \in \underline{n})}{P_1 \mid \cdots \mid P_n : \Gamma \vdash_\pi \Delta} \qquad (!) : \frac{P : \Gamma \vdash_\pi \Delta}{!\, P : \Gamma \vdash_\pi \Delta}$$

$$(pair\text{-}out) : \frac{}{\overline{a}\langle b, c\rangle : \Gamma, b{:}A \vdash_\pi a{:}A \to B, c{:}B, \Delta}\ (a, c \notin \Gamma; b \notin \Delta)$$

$$(let) : \frac{P : \Gamma, y{:}B \vdash_\pi x{:}A, \Delta}{let\langle x, y\rangle = z\ in\ P : \Gamma, z{:}A \to B \vdash_\pi \Delta}\ (x \notin \Gamma; y, z \notin \Delta)$$

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

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

Example 6.2. We can derive

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

so the following rule is derivable:

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

Notice that the rule $(pair\text{-}out)$ does not directly correspond to the logical rule $(\Rightarrow R)$, as that $(pair\text{-}in)$ does not directly correspond to $(\Rightarrow L)$; however, in view of the intended property - preservation of context assignment - this is not problematic, since we will not map rules to rules, but proofs to type derivations. This apparent discrepancy is solved by Theorem 6.7.

This notion is a true type assignment system which does not (directly) relate back to LK. For example, rules $(|)$ and $(!)$ do not change the contexts, so do not correspond to any rule in the logic, not even to a $\lambda\mu$-style [38] activation step. Moreover, rule (ν) just removes a formula, and rule $(pair\text{-}out)$ is clearly not an instance of an axiom in LK. We leave the exploration of the logical contents of this system for future work.

The following result is standard.

Lemma 6.3 (Weakening and Thinning)**.** *The following rules are admissible:*

$$(W):\ \frac{P:\Gamma \vdash_\pi \Delta}{P:\Gamma' \vdash_\pi \Delta'}\ (\Gamma' \supseteq \Gamma, \Delta' \supseteq \Delta) \qquad (T):\ \frac{P:\Gamma \vdash_\pi \Delta}{P:\Gamma' \vdash_\pi \Delta'}\ \left(\begin{array}{l}\Gamma' = \{n{:}A \in \Gamma \mid n \in fn(P)\},\\ \Delta' = \{n{:}A \in \Delta \mid n \in fn(P)\}\end{array}\right)$$

Proof. Directly from Definition 6.1. □

This result allows us to be a little less precise when we construct derivations, and allow us to freely switch to multiplicative style where rules join contexts whenever convenient. In fact, we could have defined context assignment using another approach, using the alternative rules:

$$(0):\ \frac{}{0:\varnothing \vdash_\pi \varnothing} \qquad (weak):\ \frac{P:\Gamma \vdash_\pi \Delta}{P:\Gamma' \vdash_\pi \Delta'}\ (\Gamma \subseteq \Gamma', \Delta \subseteq \Delta')$$

$$(out):\ \frac{}{\overline{a}\langle b\rangle : \beta{:}A \vdash_\pi a{:}A, b{:}A}\ (b \neq a) \qquad (|):\ \frac{P_1:\Gamma_1 \vdash_\pi \Delta_1 \quad \cdots \quad P_n:\Gamma_n \vdash_\pi \Delta_n}{P_1 \mid \cdots \mid P_n : \Gamma_1,\ldots,\Gamma_n \vdash_\pi \Delta_1,\ldots,\Delta_n}$$

$$(pair\text{-}out):\ \frac{}{\overline{a}\langle b,c\rangle : b{:}A \vdash_\pi a{:}A{\to}B, c{:}B}\ (a,c \notin \Gamma; b \notin \Delta)$$

We have a soundness (witness reduction) result, for which we first need to prove a substitution lemma and a congruence lemma.

Lemma 6.4 (Substitution)**.** *If* $P:\Gamma, x{:}A \vdash_\pi x{:}A, \Delta$ *then also* $P[b/x]:\Gamma, b{:}A \vdash_\pi b{:}A, \Delta$.

Proof. Straightforward. □

Notice that the cases $P:\Gamma \vdash_\pi n{:}A,\Delta$ and $P:\Gamma, n{:}A \vdash_\pi \Delta$ can be generalised by weakening to fit the lemma.

Lemma 6.5 (Witness congruence)**.** *If* $P:\Gamma \vdash_\pi \Delta$ *and* $P \equiv Q$, *then* $Q:\Gamma \vdash_\pi \Delta$.

Proof. By easy induction on the congruence relation. □

We now come to the main soundness result for our notion of type assignment for π.

Theorem 6.6 (Witness reduction)**.** *If* $P:\Gamma \vdash_\pi \Delta$ *and* $P \to_\pi Q$, *then* $Q:\Gamma \vdash_\pi \Delta$.

Proof. By induction on the reduction relation.

$\overline{a}\langle b\rangle \mid a(x).Q \to_\pi Q[b/x]$**:** Then the derivation is shaped like:

$$\frac{\dfrac{}{\overline{a}\langle b\rangle:\Gamma, b{:}A \vdash_\pi a{:}A, b{:}A, \Delta} \qquad \dfrac{\dfrac{\vdots}{Q:\Gamma, x{:}A \vdash_\pi x{:}A, \Delta}}{a(x).Q:\Gamma, a{:}A \vdash_\pi \Delta}}{\overline{a}\langle b\rangle \mid a(x).Q:\Gamma, a{:}A, b{:}A \vdash_\pi a{:}A, b{:}A, \Delta}$$

By Lemma 6.4, we have $Q[b/x] : \Gamma, b{:}A \vdash_\pi b{:}A, \Delta$.

$\overline{a}\langle b,c\rangle \mid a(x,y).Q \to_\pi Q[b/x,c/y]$**:** Similar.

The other cases follow by induction. $\square$

The following theorem shows that the encoding $[\![\cdot]\!]_{\mathrm{s}}$ preserves assignable types.

Theorem 6.7 (Type preservation for simple encoding)**.** *If* $P :\cdot\ \Gamma \vdash_{\mathcal{X}} \Delta$, *then* $[\![P]\!]_{\mathrm{s}} : \Gamma \vdash_\pi \Delta$.

Proof. By induction on the structure of terms in $\mathcal{X}$.

$\langle x{\cdot}\alpha\rangle$**:** Then $[\![\langle x{\cdot}\alpha\rangle]\!]_{\mathrm{s}} = x(w).\overline{\alpha}\langle w\rangle$, and the $\mathcal{X}$-derivation is shaped like:

$$\frac{}{\langle x{\cdot}\alpha\rangle : \Gamma, x{:}A \vdash_\pi \alpha{:}A, \Delta}$$

Notice that

$$\frac{\dfrac{}{\overline{\alpha}\langle w\rangle : \Gamma, w{:}A \vdash_\pi \alpha{:}A, w{:}A, \Delta}}{x(w).\overline{\alpha}\langle w\rangle : \Gamma, x{:}A \vdash_\pi \alpha{:}A, \Delta}$$

$\widehat{x}P\widehat{\alpha}{\cdot}\beta$**:** Then the $\mathcal{X}$-derivation is shaped like:

$$\frac{P : \Gamma, x{:}A \vdash_\pi \alpha{:}B, \Delta}{\widehat{x}P\widehat{\alpha}{\cdot}\beta : \Gamma \vdash_\pi \beta{:}A{\to}B, \Delta}$$

Then, by induction, $[\![P]\!]_{\mathrm{s}} : \Gamma, x{:}A \vdash_\pi \alpha{:}B, \Delta$, and we can construct:

$$\frac{\dfrac{\dfrac{\dfrac{[\![P]\!]_{\mathrm{s}} : \Gamma, x{:}A \vdash_\pi \alpha{:}B, \Delta}{!\,[\![P]\!]_{\mathrm{s}} : \Gamma, x{:}A \vdash_\pi \alpha{:}B, \Delta} \qquad \dfrac{}{\overline{\beta}\langle x,\alpha\rangle : \Gamma, x{:}A \vdash_\pi \alpha{:}B, \beta{:}A{\to}B, \Delta}}{!\,[\![P]\!]_{\mathrm{s}} \mid \overline{\beta}\langle x,\alpha\rangle : \Gamma, x{:}A \vdash_\pi \alpha{:}B, \beta{:}A{\to}B, \Delta}}{(\nu\alpha)\,(!\,[\![P]\!]_{\mathrm{s}} \mid \overline{\beta}\langle x,\alpha\rangle) : \Gamma, x{:}A \vdash_\pi \beta{:}A{\to}B, \Delta}}{(\nu x\alpha)\,(!\,[\![P]\!]_{\mathrm{s}} \mid \overline{\beta}\langle x,\alpha\rangle) : \Gamma \vdash_\pi \beta{:}A{\to}B, \Delta}$$

$P\widehat{\alpha}\,[y]\,\widehat{x}Q$**:** Then the $\mathcal{X}$-derivation is shaped like:

$$\frac{P : \Gamma \vdash_\pi \alpha{:}A, \Delta \qquad Q : \Gamma, x{:}B \vdash_\pi \Delta}{P\widehat{\alpha}\,[y]\,\widehat{x}Q : \Gamma, y{:}A{\to}B \vdash_\pi \Delta}$$

Then, by induction, we have derivations for $\llbracket P \rrbracket_{\mathrm{S}} : \Gamma \vdash_\pi \alpha{:}A, \Delta$ and $\llbracket Q \rrbracket_{\mathrm{S}} : \Gamma, x{:}B \vdash_\pi \Delta$, and we can construct:

$$\dfrac{\dfrac{\dfrac{\dfrac{\bigtriangledown}{\llbracket P \rrbracket_{\mathrm{S}} : \Gamma \vdash_\pi \alpha{:}A,\Delta}}{!\llbracket P \rrbracket_{\mathrm{S}} : \Gamma \vdash_\pi \alpha{:}A,\Delta} \quad \dfrac{\dfrac{\dfrac{}{\overline{s}\langle w\rangle : \Gamma, w{:}A \vdash_\pi s{:}A, w{:}A, \Delta}}{\alpha \rightarrow s : \Gamma, \alpha{:}A \vdash_\pi s{:}A,\Delta}}{!\,\alpha \rightarrow s : \Gamma, \alpha{:}A \vdash_\pi s{:}A,\Delta}}{\dfrac{!\llbracket P \rrbracket_{\mathrm{S}} \mid !\,\alpha \rightarrow s : \Gamma, \alpha{:}A \vdash_\pi \alpha{:}A, s{:}A, \Delta}{(\nu\alpha)\,(!\llbracket P \rrbracket_{\mathrm{S}} \mid !\,\alpha \rightarrow s) : \Gamma \vdash_\pi s{:}A,\Delta}} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\overline{x}\langle w\rangle : \Gamma, w{:}A \vdash_\pi x{:}A, w{:}A, \Delta}}{d \rightarrow x : \Gamma, d{:}B \vdash_\pi x{:}A,\Delta}}{!\,d \rightarrow x : \Gamma, d{:}B \vdash_\pi x{:}A,\Delta} \quad \dfrac{\dfrac{\bigtriangledown}{\llbracket Q \rrbracket_{\mathrm{S}} : \Gamma, x{:}B \vdash_\pi \Delta}}{!\llbracket Q \rrbracket_{\mathrm{S}} : \Gamma, x{:}B \vdash_\pi \Delta}}{\dfrac{!\,d \rightarrow x \mid !\llbracket Q \rrbracket_{\mathrm{S}} : \Gamma, d{:}B, x{:}B \vdash_\pi x{:}A,\Delta}{(\nu x)\,(!\,d \rightarrow x \mid !\llbracket Q \rrbracket_{\mathrm{S}}) : \Gamma, d{:}B \vdash_\pi \Delta}}}{\dfrac{(\nu\alpha)\,(!\llbracket P \rrbracket_{\mathrm{S}} \mid !\,\alpha \rightarrow s) \mid (\nu x)\,(!\,d \rightarrow x \mid !\llbracket Q \rrbracket_{\mathrm{S}}) : \Gamma, d{:}B \vdash_\pi s{:}A, \Delta}{y(s,d).\,((\nu\alpha)\,(!\llbracket P \rrbracket_{\mathrm{S}} \mid !\,\alpha \rightarrow s) \mid (\nu x)\,(!\,d \rightarrow x \mid !\llbracket Q \rrbracket_{\mathrm{S}})) : \Gamma, y{:}A \rightarrow B \vdash_\pi \Delta}}$$

$P\widehat{\alpha} \dagger \widehat{x}Q$**:** Then the $\mathcal{X}$-derivation is shaped like:

$$\dfrac{\dfrac{\bigtriangledown}{P : \Gamma \vdash_\pi \alpha{:}A, \Delta} \quad \dfrac{\bigtriangledown}{Q : \Gamma, x{:}A \vdash_\pi \Delta}}{P\widehat{\alpha} \dagger \widehat{x}Q : \Gamma \vdash_\pi \Delta}$$

By induction, we have derivations for both $\llbracket P \rrbracket_{\mathrm{S}} : \Gamma \vdash_\pi \alpha{:}A, \Delta$ and $\llbracket Q \rrbracket_{\mathrm{S}} : \Gamma, x{:}A \vdash_\pi \Delta$. Then we can construct:

$$\dfrac{\dfrac{\dfrac{\bigtriangledown}{\llbracket P \rrbracket_{\mathrm{S}} : \Gamma \vdash_\pi \alpha{:}A,\Delta}}{!\llbracket P \rrbracket_{\mathrm{S}} : \Gamma \vdash_\pi \alpha{:}A,\Delta} \quad \dfrac{\dfrac{\dfrac{}{\overline{x}\langle w\rangle : \Gamma, w{:}A \vdash_\pi x{:}A, w{:}A, \Delta}}{\alpha \rightarrow x : \Gamma, \alpha{:}A \vdash_\pi x{:}A, \Delta}}{!\,\alpha \rightarrow x : \Gamma, \alpha{:}A \vdash_\pi x{:}A, \Delta} \quad \dfrac{\dfrac{\bigtriangledown}{\llbracket Q \rrbracket_{\mathrm{S}} : \Gamma, x{:}A \vdash_\pi \Delta}}{!\llbracket Q \rrbracket_{\mathrm{S}} : \Gamma, x{:}A \vdash_\pi \Delta}}{\dfrac{!\llbracket P \rrbracket_{\mathrm{S}} \mid !\,\alpha \rightarrow x \mid !\llbracket Q \rrbracket_{\mathrm{S}} : \Gamma, \alpha{:}A, x{:}A \vdash_\pi \alpha{:}A, x{:}A, \Delta}{\dfrac{(\nu x)\,(!\llbracket P \rrbracket_{\mathrm{S}} \mid !\,\alpha \rightarrow x \mid !\llbracket Q \rrbracket_{\mathrm{S}}) : \Gamma, \alpha{:}A \vdash_\pi \alpha{:}A, \Delta}{(\nu \alpha x)\,(!\llbracket P \rrbracket_{\mathrm{S}} \mid !\,\alpha \rightarrow x \mid !\llbracket Q \rrbracket_{\mathrm{S}}) : \Gamma \vdash_\pi \Delta}}}$$

□

We can also show that the encoding $\llbracket \cdot \rrbracket_{\mathrm{F}}$ preserves assignable types.

Theorem 6.8 (Type preservation for full encoding)**.** *If* $P :\cdot \Gamma \vdash_{\mathcal{X}} \Delta$, *then* $\llbracket P \rrbracket_{\mathrm{F}} : \Gamma \vdash_\pi \Delta$.

Proof. Since $\llbracket \cdot \rrbracket_{\mathrm{S}}$ and $\llbracket \cdot \rrbracket_{\mathrm{F}}$ differ only in the interpretation of *import*, we only need to check that case.

$P\widehat{\alpha}\,[y]\,\widehat{x}Q$**:** Then the $\mathcal{X}$-derivation is shaped like:

$$\dfrac{\dfrac{\bigtriangledown}{P : \Gamma \vdash_\pi \alpha{:}A, \Delta} \quad \dfrac{\bigtriangledown}{Q : \Gamma, x{:}B \vdash_\pi \Delta}}{P\widehat{\alpha}\,[y]\,\widehat{x}Q : \Gamma, y{:}A \rightarrow B \vdash_\pi \Delta}$$

Then, by induction, we have derivations for $\llbracket P \rrbracket_{\mathrm{F}} : \Gamma \vdash_{\pi} \alpha{:}A, \Delta$ and $\llbracket Q \rrbracket_{\mathrm{F}} : \Gamma, x{:}B \vdash_{\pi} \Delta$, and we can construct:

$$\dfrac{\dfrac{\dfrac{\nabla}{\llbracket P \rrbracket_{\mathrm{S}} : \Gamma \vdash_{\pi} \alpha{:}A, \Delta}}{!\llbracket P \rrbracket_{\mathrm{S}} : \Gamma \vdash_{\pi} \alpha{:}A, \Delta} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{v}\langle w\rangle : \Gamma, w{:}A \vdash_{\pi} v{:}A, w{:}A, \Delta}}{\alpha{\to}v : \Gamma, \alpha{:}A \vdash_{\pi} v{:}A, \Delta}}{!\alpha{\to}v : \Gamma, \alpha{:}A \vdash_{\pi} v{:}A, \Delta} \quad \dfrac{\dfrac{\dfrac{}{\overline{x}\langle w\rangle : \Gamma, w{:}B \vdash_{\pi} x{:}B, w{:}B, \Delta}}{d{\to}x : \Gamma, d{:}B \vdash_{\pi} x{:}B, \Delta}}{!d{\to}x : \Gamma, d{:}B \vdash_{\pi} x{:}B, \Delta}}{!\alpha{\to}v \mid !d{\to}x : \Gamma, \alpha{:}A, d{:}B \vdash_{\pi} x{:}B, v{:}A, \Delta}}{y(v,d).(!\alpha{\to}v \mid !d{\to}x) : \Gamma, \alpha{:}A, y{:}A{\to}B \vdash_{\pi} x{:}B, \Delta} \quad \dfrac{\dfrac{\nabla}{\llbracket Q \rrbracket_{\mathrm{S}} : \Gamma, x{:}B \vdash_{\pi} \Delta}}{!\llbracket Q \rrbracket_{\mathrm{S}} : \Gamma, x{:}B \vdash_{\pi} \Delta}}{\dfrac{!\llbracket P \rrbracket_{\mathrm{F}} \mid y(v,d).(!\alpha{\to}v \mid !d{\to}x) \mid !\llbracket Q \rrbracket_{\mathrm{F}} : \Gamma, x{:}B, \alpha{:}A, y{:}A{\to}B \vdash_{\pi} x{:}B, \alpha{:}A, \Delta}{\dfrac{(\nu x)\,(!\llbracket P \rrbracket_{\mathrm{F}} \mid y(v,d).(!\alpha{\to}v \mid !d{\to}x) \mid !\llbracket Q \rrbracket_{\mathrm{F}}) : \Gamma, \alpha{:}A, y{:}A{\to}B \vdash_{\pi} \alpha{:}A, \Delta}{(\nu\alpha x)\,(!\llbracket P \rrbracket_{\mathrm{F}} \mid y(v,d).(!\alpha{\to}v \mid !d{\to}x) \mid !\llbracket Q \rrbracket_{\mathrm{F}}) : \Gamma, y{:}A{\to}B \vdash_{\pi} \Delta}}}$$

Notice that $(\nu\alpha x)\,(!\llbracket P \rrbracket_{\mathrm{F}} \mid y(v,d).(!\alpha{\to}v \mid !d{\to}x) \mid !\llbracket Q \rrbracket_{\mathrm{F}}) = \llbracket P\widehat{\alpha}\,[y]\,\widehat{x}Q \rrbracket_{\mathrm{F}}$. ☐

We can show this result for the replicative encoding as well.

Theorem 6.9 (Type preservation for replicative encoding)**.** *If* $P :\cdot \Gamma \vdash_{\mathcal{X}} \Delta$*, then* $\llbracket P \rrbracket_{\mathrm{R}} : \Gamma \vdash_{\pi} \Delta$.

Proof. By induction on the structure of terms in $\mathcal{X}$. Since $\llbracket\cdot\rrbracket_{\mathrm{R}}$ differs from $\llbracket\cdot\rrbracket_{\mathrm{F}}$ only in the use of replication, and the rule $(!)$ does not change contexts, the proof is much the same as the one for Theorem 6.7. ☐

7. Expressing Negation

In this section we will look at the logical connective $\neg$ and how to encode it in the π-calculus.

Definition 7.1. The sequent rules that correspond to negation are as follows:

$$(\neg R) : \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta} \qquad (\neg L) : \frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}$$

To extend the Curry-Howard isomorphism of $\mathcal{X}$ also to these connectors, we follow the same approach as used for the arrow: a disappearing formula in a context corresponds to a connector that gets bound, and a formula that appears in a context corresponds to a connector that is introduced.

Definition 7.2. We extend $\mathcal{X}$'s syntax with the following constructs:

$$\begin{array}{rcll} P & ::= & \ldots \mid x\cdot P\widehat{\alpha} & \textit{left inversion} \\ & & \mid \widehat{x}P\cdot\alpha & \textit{right inversion} \end{array}$$

Definition 7.3. We extend the set of types by

$$A, B \ ::= \ \cdots \mid \neg A$$

(as usual, $\neg$ binds stronger than $\to$) and add the type assignment rules:

$$(\mathit{inv\text{-}r}) : \frac{P :\cdot \Gamma, x{:}A \vdash_{\mathcal{X}} \Delta}{\widehat{x}P\cdot\alpha :\cdot \Gamma \vdash_{\mathcal{X}} \alpha{:}\neg A, \Delta} \qquad (\mathit{inv\text{-}l}) : \frac{P :\cdot \Gamma \vdash_{\mathcal{X}} \alpha{:}A, \Delta}{x\cdot P\widehat{\alpha} :\cdot \Gamma, x{:}\neg A \vdash_{\mathcal{X}} \Delta}$$

For example, we can show

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle y{\cdot}\alpha\rangle \mathrel{:\cdot} y{:}A \vdash_{\mathcal{X}} \alpha{:}A}}{\widehat{y}\langle y{\cdot}\alpha\rangle{\cdot}\gamma \mathrel{:\cdot} \;\vdash_{\mathcal{X}} \gamma{:}\neg A, \alpha{:}A}}{x\cdot(\widehat{y}\langle y{\cdot}\alpha\rangle{\cdot}\gamma)\widehat{\gamma} \mathrel{:\cdot} x{:}\neg\neg A \vdash_{\mathcal{X}} \alpha{:}A}}{\widehat{x}\,(x\cdot(\widehat{y}\langle y{\cdot}\alpha\rangle{\cdot}\gamma)\widehat{\gamma})\widehat{\alpha}{\cdot}\beta \mathrel{:\cdot} \;\vdash_{\mathcal{X}} \beta{:}\neg\neg A{\to}A}$$

The notion of reduction is extended naturally by adding the following reduction rules.

Definition 7.4. The logical rule for negation is:

$$(\widehat{y}P{\cdot}\beta)\widehat{\beta} \dagger \widehat{x}\,(x{\cdot}Q\widehat{\alpha}) \;\to\; Q\widehat{\alpha} \dagger \widehat{y}P$$

We extend the notion of introduced connector by saying that also $P = x{\cdot}Q\widehat{\alpha}$ with $x \notin fs(Q)$ introduces x, and $P = \widehat{x}Q{\cdot}\alpha$ with $\alpha \notin fp(Q)$ introduces α. We add the propagation rules:

$$\begin{array}{rcll}
(y{\cdot}Q\widehat{\beta})\widehat{\alpha} \not\nearrow \widehat{x}P & \to & y{\cdot}(Q\widehat{\alpha} \not\nearrow \widehat{x}P)\widehat{\beta} & \\
(\widehat{y}Q{\cdot}\beta)\widehat{\alpha} \not\nearrow \widehat{x}P & \to & \widehat{y}(Q\widehat{\alpha} \not\nearrow \widehat{x}P){\cdot}\beta & \alpha \neq \beta \\
(\widehat{y}Q{\cdot}\alpha)\widehat{\alpha} \not\nearrow \widehat{x}P & \to & (\widehat{y}(Q\widehat{\alpha} \not\nearrow \widehat{x}P){\cdot}\beta)\widehat{\beta} \dagger \widehat{x}P & \\[6pt]
P\widehat{\alpha} \not\nwarrow \widehat{x}(y{\cdot}Q\widehat{\beta}) & \to & y{\cdot}(P\widehat{\alpha} \not\nwarrow \widehat{x}Q)\widehat{\beta} & x \neq y \\
P\widehat{\alpha} \not\nwarrow \widehat{x}(x{\cdot}Q\widehat{\beta}) & \to & P\widehat{\alpha} \not\nwarrow \widehat{y}\,(y{\cdot}(P\widehat{\alpha} \not\nwarrow \widehat{x}Q)\widehat{\beta}) & \\
P\widehat{\alpha} \not\nwarrow \widehat{x}(\widehat{y}Q{\cdot}\beta) & \to & \widehat{y}(P\widehat{\alpha} \not\nwarrow \widehat{x}Q){\cdot}\beta &
\end{array}$$

Notice that now we have *cuts* that do not contract, as

$$(\widehat{y}Q\widehat{\gamma}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}\,(x{\cdot}P\widehat{\beta})$$

where $\alpha \notin fp(Q)$, and $x \notin fs(P)$, since none of the rules are applicable; however, *typeable cuts* do contract.

We will now extend the three encodings so that we deal with the added connective as well.

Definition 7.5 (Negation)**.** Negation gets represented in the π-calculus via the simple encoding as:

$$[\![x{\cdot}P\widehat{\alpha}]\!]_{\mathrm{S}} \;=\; x(z).\,((\nu\alpha)\,(!\,[\![P]\!]_{\mathrm{F}} \mid !\,\alpha{\to}z))$$

via the full encoding as:

$$\begin{array}{rcl}
[\![x{\cdot}P\widehat{\alpha}]\!]_{\mathrm{F}} & = & (\nu\alpha)\,(!\,[\![P]\!]_{\mathrm{F}} \mid x(z).\,(!\,\alpha{\to}z)) \\
[\![\widehat{x}P{\cdot}\alpha]\!]_{\mathrm{F}} & = & (\nu x)\,(!\,[\![P]\!]_{\mathrm{F}} \mid \overline{\alpha}\langle x\rangle)
\end{array}$$

and via the replicative encoding as:

$$\begin{array}{rcl}
[\![x{\cdot}P\widehat{\alpha}]\!]_{\mathrm{R}} & = & (\nu\alpha)\,([\![P]\!]_{\mathrm{R}} \mid !\,x(z).\,(!\,\alpha{\to}z)) \\
[\![\widehat{x}P{\cdot}\alpha]\!]_{\mathrm{R}} & = & (\nu x)\,([\![P]\!]_{\mathrm{R}} \mid !\,\overline{\alpha}\langle x\rangle)
\end{array}$$

This encoding of inversion explains the role of negation in detail. If P is outputting on α, but no connection to α is available, input is needed from a process Q that will send one of its input names z. Once received, P can output on α which gets connected to z; so Q will provide a means for P to continue, and is therefore aptly called a *continuation*.

The full encoding of the witness for $\neg\neg A{\to}A$ now becomes:

$$\begin{array}{l}
[\![\widehat{x}\,(x\cdot(\widehat{y}\langle y{\cdot}\alpha\rangle{\cdot}\gamma)\widehat{\gamma})\widehat{\alpha}{\cdot}\beta]\!]_{\mathrm{F}} \hspace{4cm} = \\
(\nu x\alpha)\,(!\,(\nu\gamma)\,(!\,(\nu y)\,(!\,y(w).\,\overline{\alpha}\langle w\rangle \mid \overline{\gamma}\langle y\rangle) \mid x(z).\,(!\,\gamma{\to}z)) \mid \overline{\beta}\langle x,\alpha\rangle)]
\end{array}$$

The following consistency result is now easy to prove.

Lemma 7.6. *$\llbracket (\widehat{y}P\cdot\beta)\widehat{\beta} \dagger \widehat{x}(x\cdot Q\widehat{\alpha}) \rrbracket_{\mathsf{F}} \;{}_{\mathsf{c}}\!\sqsupseteq\; \llbracket Q\widehat{\alpha} \dagger \widehat{y}P \rrbracket_{\mathsf{F}}$, if β, x introduced.*

Proof.
$$\begin{array}{lll}
\llbracket (\widehat{y}P\cdot\beta)\widehat{\beta} \dagger \widehat{x}(x\cdot Q\widehat{\alpha}) \rrbracket_{\mathsf{F}} & \triangleq & \\
(\nu\beta x)\,(!\,(\nu y)\,(!\,\llbracket P \rrbracket_{\mathsf{F}} \mid \overline{\beta}\langle y\rangle) \mid !\,\beta{\rightarrow}x \mid !\,(\nu\alpha)\,(!\,\llbracket Q \rrbracket_{\mathsf{F}} \mid x(z).\,(!\,\alpha{\rightarrow}z))) & {}_{\mathsf{c}}\!\sqsupseteq & \\
(\nu\beta x)\,((\nu y)\,(!\,\llbracket P \rrbracket_{\mathsf{F}} \mid \overline{\beta}\langle y\rangle) \mid \beta{\rightarrow}x \mid (\nu\alpha)\,(!\,\llbracket Q \rrbracket_{\mathsf{F}} \mid x(z).\,(!\,\alpha{\rightarrow}z))) & \rightarrow_{\pi} & (\beta, x) \\
(\nu\alpha y)\,(!\,\llbracket Q \rrbracket_{\mathsf{F}} \mid !\,\alpha{\rightarrow}y \mid !\,\llbracket P \rrbracket_{\mathsf{F}}) & \triangleq & \llbracket Q\widehat{\alpha} \dagger \widehat{y}P \rrbracket_{\mathsf{F}}
\end{array}$$

The correctness for the other two encodings follows in a similar way, and that of the propagation rules follows as above in Theorem 5.3.

We add the following type assignment rules for negation:

Definition 7.7 (Type assignment rules in $\vdash_{\pi}$ for $\neg$)**.**

$$(\textit{inv-r}) : \frac{}{\overline{a}\langle x\rangle : \Gamma, x{:}A \vdash_{\pi} a{:}\neg A, \Delta}\,(x \notin \Delta) \qquad (\textit{inv-l}) : \frac{P : \Gamma \vdash_{\pi} a{:}A, \Delta}{x(a).\,P : \Gamma, x{:}\neg A \vdash_{\pi} \Delta}\,(a \notin \Gamma)$$

We can now check that the extended encoding preserves assignable types as well.

Theorem 7.8. *If $P \mathrel{:\!\cdot} \Gamma \vdash_{\mathcal{X}} \Delta$, then $\llbracket P \rrbracket_{\mathsf{F}} : \Gamma \vdash_{\pi} \Delta$.*

Proof. By induction on the structure of of terms in $\mathcal{X}$; we only show the two added cases to the proof of Theorem 6.7.

$x\cdot P\widehat{\alpha}$**:** Then the last rule applied in the $\mathcal{X}$-derivation is $(\neg L)$:

$$\frac{P \mathrel{:\!\cdot} \Gamma \vdash_{\mathcal{X}} \alpha{:}A, \Delta}{x\cdot P\widehat{\alpha} \mathrel{:\!\cdot} \Gamma, x{:}\neg A \vdash_{\mathcal{X}} \Delta}$$

and, by induction, $\llbracket P \rrbracket_{\mathsf{F}} : \Gamma \vdash_{\pi} \alpha{:}A, \Delta$, and we can construct:

$$\dfrac{\dfrac{\llbracket P \rrbracket_{\mathsf{F}} : \Gamma \vdash_{\pi} \alpha{:}A, \Delta}{!\,\llbracket P \rrbracket_{\mathsf{F}} : \Gamma \vdash_{\pi} \alpha{:}A, \Delta}\,(|) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\overline{z}\langle w\rangle : \Gamma, w{:}A \vdash_{\pi} z{:}A, w{:}A}\,(\textit{out})}{\alpha{\rightarrow}z : \Gamma, \alpha{:}A \vdash_{\pi} z{:}A, \Delta}\,(\textit{in})}{!\,\alpha{\rightarrow}z : \Gamma, \alpha{:}A \vdash_{\pi} z{:}A, \Delta}\,(!)}{x(z).\,(!\,\alpha{\rightarrow}z) : \Gamma, \alpha{:}A, x{:}\neg A \vdash_{\pi} \Delta}\,(\textit{inv-l})}{\dfrac{!\,\llbracket P \rrbracket_{\mathsf{F}} \mid x(z).\,(!\,\alpha{\rightarrow}z) : \Gamma, \alpha{:}A, x{:}\neg A \vdash_{\pi} \alpha{:}A, \Delta}{(\nu\alpha)\,(!\,\llbracket P \rrbracket_{\mathsf{F}} \mid x(z).\,(!\,\alpha{\rightarrow}z)) : \Gamma, x{:}\neg A \vdash_{\pi} \Delta}\,(\varphi)}\,(|)$$

$\widehat{x}P\cdot\alpha$**:** Then the last rule applied in the $\mathcal{X}$-derivation is $(\neg R)$:

$$\frac{P \mathrel{:\!\cdot} \Gamma, x{:}A \vdash_{\mathcal{X}} \Delta}{\widehat{x}P\cdot\alpha \mathrel{:\!\cdot} \Gamma \vdash_{\mathcal{X}} \alpha{:}\neg A, \Delta}$$

and, by induction, $[\![P]\!]_{\mathrm{F}} : \Gamma, x{:}A \vdash_\pi \Delta$, and we can construct:

$$\dfrac{\dfrac{\dfrac{\vdots}{[\![P]\!]_{\mathrm{F}} : \Gamma, x{:}A \vdash_\pi \Delta}}{!\,[\![P]\!]_{\mathrm{F}} : \Gamma, x{:}A \vdash_\pi \Delta}\,(!) \qquad \dfrac{}{\overline{\alpha}\langle x\rangle : \Gamma, x{:}A \vdash_\pi \alpha{:}\neg A, \Delta}\,(\mathit{inv\text{-}r})}{\dfrac{!\,[\![P]\!]_{\mathrm{F}} \mid \overline{\alpha}\langle x\rangle : \Gamma, x{:}A \vdash_\pi \alpha{:}\neg A, \Delta}{(\nu x)\,(!\,[\![P]\!]_{\mathrm{F}} \mid \overline{\alpha}\langle x\rangle) : \Gamma \vdash_\pi \alpha{:}\neg A, \Delta}\,(\nu)}\,(\mid)$$

so our extended encoding respects the classical sequent logic rules.

CONCLUSIONS

In this paper we have bridged the gap between the computational content of *cut*-elimination and the semantics of concurrent calculi, by presenting encodings of Gentzen's classical sequent calculus LK to the π-calculus that preserve *cut*-elimination. This was achieved through an embedding of the calculus $\mathcal{X}$ into the π-calculus that implements a *cut* as communication. $\mathcal{X}$'s terms directly represent proofs in LK, by naming assumptions with Roman characters, and conclusions with Greek characters, and seeing these as *input* and *output*, respectively, but terms in $\mathcal{X}$ can also not correspond to proofs; $\mathcal{X}$' introduces a simple concept of input and output that naturally translates into the input and output primitives of the π-calculus.

The main operative of $\mathcal{X}$, the *cut*, gets represented by $P\widehat{\alpha} \dagger \widehat{x}Q$, and we interpret this term in the π-calculus as a communication: we see P as a process that outputs over α, and Q as a process that inputs over x, and communication between these terms uses the *forwarder* $\alpha(w).\overline{x}\langle w\rangle$. To make sure that the correct communication takes place we make use of the mobility feature of the $\pi-$calculus *i.e.* private names are sent to the communicating party and used for later communication as channel names.

We presented three different encodings, each with specific interesting properties. We first presented the simple encoding, and showed that it preserves $\mathcal{X}$'s head-reduction; in this encoding we cannot represent full *cut*-elimination because we place some interpreted terms under *input*, in particular when encoding the witness for $(\rightarrow L)$. This seems to be a natural consequence, and is a feature also in the encoding of the λ-calculus [36, 41, 13]; while this initial result is interesting, the important question to answer is whether full-cut elimination can encoded.

In fact, we have shown in this paper that the limitation of *input* can easily be avoided. To that purpose, we introduced the concept of *synchronisation cell*, and managed to show that, by slightly modifying our encoding, we could represents full *cut*-elimination. The third encoding is more abstract, and interprets terms as infinite resources which simplifies the proofs.

By our result, we have shown that the π-calculus is a fully expressive model of computation, whereby we extend the results of Milner's seminal paper [36] and others (see [41]); using our new approach, we are capable of not just encoding lazy reduction for the λ-calculus (as in those papers) or spine reduction as in [13], but can treat reduction in full. And, in fact, this approach can be extended to the λ-calculus as well, as well as to $\overline{\lambda}\mu\tilde{\mu}$ [12]. Through this result, we have shown that the π-calculus is fully expressive in that it is not only possible to represent the functional paradigm, but can also represents both *context call* and *parameter call* (as expressed in $\mathcal{X}$ via, respectively, left and right propagation) in full via representing proofs and proof contractions in LK.

The variant of the π-calculus we considered uses a pairing facility which enables the definition of a notion of implicative type assignment on processes. Using this notion, we proved that proofs in

LK have a representation in π; our *cut*-elimination results then show that not only do we correctly represent reduction on the calculus $\mathcal{X}$, but also can model proofs in LK in all detail in such a way that *cut*-elimination is preserved by contextual equivalence. We also represented negation in $\mathcal{X}$ by extending the syntax and reduction rules, and extended our encodings to deal with the added constructs; we have shown that all representation results still hold; since we have successfully represented both implication and negation, this implies that this can then easily be extended to conjunction and disjunction.

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. Computational Interpretations of Linear Logic. *Theoretical Computer Science*, 111(1&2):3–57, 1993.

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

[6] S. Abramsky and R. Jagadeesan. Games and Full Completeness for Multiplicative Linear Logic. *Journal of Symbolic Logic*, 59(2):543–574, 1994.

[7] Z. M. Ariola and H. Herbelin. Minimal Classical Logic and Control Operators. In *ICALP*, pages 871–885, 2003.

[8] 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.

[9] S. van Bakel, S. Lengrand, and P. Lescanne. The language $\mathcal{X}$: circuits, computations and Classical Logic. In Mario Coppo, Elena Lodi, and G. Michele Pinna, editors, *Proceedings of Ninth Italian Conference on Theoretical Computer Science* (ICTCS'05), *Siena, Italy*, volume 3701 of *Lecture Notes in Computer Science*, pages 81–96. Springer Verlag, 2005.

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

[11] S. van Bakel and J. Raghunandan. Capture Avoidance and Garbage Collection for $\mathcal{X}$. Presented at the Third International Workshop on *Term Graph Rewriting 2006* (TermGraph'06), Vienna, Austria, 2006.

[12] S. van Bakel and M.G. Vigliotti. The π-calculus as a Universal Model of Computation. Submitted.

[13] 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.

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

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

[16] 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.

[17] N. G. de Bruijn. A namefree lambda calculus with facilities for internal definition of expressions and segments. TH-Report 78-WSK-03, Technological University Eindhoven, Netherlands, Department of Mathematics, 1978.

[18] Paola Bruscoli. A purely logical account of sequentiality in proof search. In Peter 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.

[19] Paola Bruscoli and Alessio 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.

[20] M. Cimini, C. Sacerdoti Coen, and D. Sangiorgi. $\overline{\lambda}\mu\tilde{\mu}$ calculus, π-calculus, and abstract machines. In *Proceedings of EXPRESS'09*, 2009.

[21] T. Coquand and G. Huet. The Calculus of Constructions. *Information and Computation*, 76(2,3):95–120, 1988.

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

[23] H.B. Curry and R. Feys. *Combinatory Logic*, volume 1. North-Holland, Amsterdam, 1958.

[24] G. Gentzen. Untersuchungen über das Logische Schliessen. *Mathematische Zeitschrift*, 39:176–210 and 405–431, 1935.

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

[26] J.-Y. Girard. A new constrcutive logic: classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, 1991.

[27] J.Y. Girard. The System F of Variable Types, Fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986.

[28] T. Griffin. A formulae-as-types notion of control. In *Proceedings of the 17th Annual ACM Symposium on Principles Of Programming Languages, Orlando (Fla., USA)*, pages 47–58, 1990.

[29] H. Herbelin. *Séquents qu'on calcule : de l'interprétation du calcul des séquents comme calcul de λ-termes et comme calcul de stratégies gagnantes*. Thèse d'université, Université Paris 7, Janvier 1995.

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

[31] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In *Proceedings of ECOOP'91*, volume 512 of *Lecture Notes in Computer Science*, pages 133–147. Springer Verlag, 1991.

[32] K. Honda and N. Yoshida. On the Reduction-based Process Semantics. *Theoretical Computer Science*, 151:437–486, 1995.

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

[34] S.C. Kleene. *Introduction to Metamathematics*. Études et Recherches en Informatique. North Holland, Amsterdam, 1952.

[35] J.W. Klop. Term Rewriting Systems. In S. Abramsky, Dov.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 1, pages 1–116. Clarendon Press, 1992.

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

[37] R. Milner. The Polyadic π-Calculus: A Tutorial. In F.L Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification*. Springer Verlag, Secaucus, NJ, USA, 1993.

[38] M. Parigot. An algorithmic interpretation of classical natural deduction. In *Proceedings of Int. Conf. on Logic Programming and Automated Reasoning* (LPAR'92), volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer Verlag, 1992.

[39] Benjamin C. Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–453, 1996.

[40] D. Sangiorgi. Lazy functions and processes. Rapport de Recherche 2515, INRIA, Sophia-Antipolis, France, 1995.

[41] D. Sangiorgi and D. Walker. On barbed equivalences in the π-calculus. In *Proceedings of CONCUR'01*, volume 2154 of *Lecture Notes in Computer Science*, 2001.

[42] A.J. Summers. Approaches to Polymorphism in Classical Sequent Calculus. Manuscript, 2008.

[43] A.J. Summers. *Curry-Howard Term Calculi for Gentzen-Style Classical Logic*. PhD thesis, Imperial College London, 2008.

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

[45] C. Urban. *Classical Logic and Computation*. PhD thesis, University of Cambridge, October 2000.

[46] C. Urban. Strong Normalisation for a Gentzen-like Cut-Elimination Procedure. In *Proceedings of Typed Lambda Calculus and Applications (TLCA'01)*, volume 2044 of *Lecture Notes in Computer Science*, pages 415–429, 2001.

[47] C. Urban and G. M. Bierman. Strong normalisation of cut-elimination in classical logic. *Fundamenta Informaticae*, 45(1,2):123–155, 2001.

[48] P. Wadler. Call-by-Value is Dual to Call-by-Name. In *Proceedings of the eighth ACM SIGPLAN international conference on Functional programming*, pages 189 – 201, 2003.