Classical Cut-elimination in the π-calculus

(In memory of Kohei Honda)

(*Under submission*)

Steffen van Bakel, Luca Cardelli, and Maria Grazia Vigliotti

[1] Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ, UK
[2] Microsoft Research Cambridge, 7 J J Thomson Avenue, Cambridge, CB3 0FB, UK
[3] Rail Safety and Standards Board, Block 2, Angel Square, 1 Torrens Street, London EC1V 1NY

svb@doc.ic.ac.uk, luca@microsoft.com, mgv98@doc.ic.ac.uk

Abstract

We define the calculus $\mathcal{LK}$ - a variant of the calculus $\mathcal{X}$ - that enjoys the Curry-Howard correspondence for Gentzen's calculus LK; the variant consists of allowing arbitrary progress of *cut* over *cut*. We study the π-calculus enriched with pairing, for which we define a notion of implicative type assignment. We translate the terms of $\mathcal{LK}$ into this variant of π, and show that reduction and assignable types are preserved. This implies that all proofs in LK have a representation in π, and that *cut*-elimination is effectively simulated by π's synchronisation, congruence, and bisimilarity between processes.

We present two interpretations for which we show soundness results (but with respect to different notions of reduction), as well as type preservation. Using the second interpretation, we show that we preserve Gentzen's *Hauptsatz* result, and prove completeness.

We then enrich the logic with the connector $\neg$ (negation), and show that this also can be represented in π, whilst preserving the results.

keywords: classical logic, sequent calculus, pi calculus, translation, type assignment

Introduction

In this paper we present two translations of proofs of Gentzen's (implicative) proof calculus for Classical Logic LK [28] into the π-calculus [42] that respect *cut*-elimination. These translations are attained through using the intuition of the calculus $\mathcal{X}$, which gives a computational content to LK (a first version of this calculus was proposed in [46, 48, 47]; the implicative fragment of $\mathcal{X}$ was studied in [9]). We will here use a variant of $\mathcal{X}$, called $\mathcal{LK}$ – obtained by not using $\mathcal{X}$'s activated cuts but allowing arbitrary *cut*-over-*cut* reduction – which satisfies most properties shown to hold for $\mathcal{X}$ (with the exception of strong normalisation, but this is as expected for any calculus that models full *cut*-elimination).

$\mathcal{LK}$ 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 $\mathcal{LK}$ are different from those in other calculi used for logic in that they have multiple named inputs and multiple named outputs, that are collectively called *connectors*. Reduction in $\mathcal{LK}$ 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.

The two main features of $\mathcal{X}$ –non-confluence and reduction as (re-)connection of terms via the exchange of names– are also manifest in the π-calculus, an observation which inspired us to consider the π-calculus as a means to model *cut*-elimination and proofs in LK. The aim of this paper is to link LK and π via $\mathcal{LK}$; we achieve this through the definition of two different translations that map (untyped) $\mathcal{LK}$-terms to π-processes.

Main considerations

In setting up the translations we present here, some difficulties had to be overcome.

The first is that of non-confluence: it is well known that *cut*-elimination in LK is not confluent, and, since $\mathcal{LK}$ is Curry-Howard for LK and its reduction respects *cut*-elimination, neither is reduction in $\mathcal{LK}$. Moreover, the calculus is symmetric, where call-by-value and call-by-name coexists as sub-reduction systems that differ in how cuts are evaluated. A *cut* $P\widehat{\alpha} \dagger \widehat{x}Q$ in $\mathcal{LK}$ (see Definition 1.1) expresses two terms that need to be connected via α and x, which can be realised both through the substitution of Q for α in P (which is essentially part of CBV) as well as through that of P for x in Q (part of CBN), which normally give different results: 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)

An important difference between $\mathcal{LK}$ and the π-calculus is that $\mathcal{LK}$ has explicit duplication of terms through reduction rules, whereas in the π-calculus this can only be achieved through replication, effectively “flooding the system.” If we model P and Q in π through $[\![\cdot]\!]$, then we obtain one process sending on α, and one receiving on x (we can link these via $\alpha(w).\overline{x}\,w$). 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; this implies that the translation of the cut has to use replication for both sub-terms (see Definition 4.1 and 5.1).

As a consequence of this abundance of replication, we cannot simply show that the reduction of the interpretation of a *cut* runs to the interpretation of its contractum; for example, when the reduction of $[\![P\widehat{\alpha} \dagger \widehat{x}Q]\!]$ substitutes $[\![Q]\!]$ for α in $[\![P]\!]$, only one copy of $[\![P]\!]$ is used, leaving $![\![P]\!]$ which can generate superfluous observable behaviour. Therefore, we will rather show that the interpretation of the contractum of a *cut* has *less* observable behaviour than the interpretation of the *cut* itself. This is, in fact, common for interpretations of non-confluent calculi , where the interpretation of terms decreases under reduction (see Remark 2.7).

The second point is that we aim for our interpretation to be meaningful also from the point of view of the logic, and would like to not only link LK and the π-calculus as systems of reduction, but also a systems of proofs and have the interpretation of terms respect the assignable types. In this precise sense we aim to view processes in π as giving an alternative (computational) meaning to proofs in classical logic. The problem there is that there does not exist a notion of type assignment for the π-calculus that naturally deals with function (arrow) types. Some attempts have been made, for example by linearising the calculus, but these seem very much to be based on the principle of type assignment as can be found in the λ-calculus and its siblings: if $\Gamma \vdash_{\lambda} M : A$, where A is the type of the ‘result’ of M, we can construct an abstraction over M towards *any* of its free variables x, which occurs in Γ with a type B, but *only* to its result type A to derive $\Gamma \setminus \{x{:}B\} \vdash_{\lambda} \lambda x.M : B \to A$. However, this is not the right approach for the π-calculus, since a process does not have a unique type itself since not a unique result, but rather has many types associated to it through its outputs.

We see this same feature in $\mathcal{LK}$, where a term is associated to two contexts through $P : \Gamma \vdash_{\text{LK}} \Delta$, with Γ containing the types for its inputs, and Δ for its outputs. Now in the rule (exp), where the term representation for right-introduction of the arrow is typed,

$$\frac{P : \Gamma, x{:}A \vdash \alpha{:}B, \Delta}{\widehat{x}P\widehat{\alpha}\cdot\beta : \Gamma \vdash \beta{:}A{\rightarrow}B, \Delta}$$

any type in the left context can be paired to any type in the right context to introduce an arrow type, essentially not just binding an input, but also an output in *one go*. Since in $\mathcal{LK}$ 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, the 'classical logic' point of view is that P is 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. This case is interesting in that it highlights a special feature of $\mathcal{LK}$, not found in other calculi, which is the *simultaneous binding of two free names*.

Perhaps surprisingly, we can do something similar for the π-calculus to define a natural notion of functional type assignment: since processes have in general multiple input and output channels, it is natural from a classical logic point of view to see a process P as an entity over which we can construct functions by picking (binding) *both* input and output. Since we now 'capture' two channel names in one action, we have to introduce a notion of pairing to the π-calculus, and define $\pi_{\langle\rangle}$, the π-calculus is extended with pairing [2] (see Definition 3.1); we can naturally assign arrow types to channel names over which pairs are transmitted, and come to the notion of implicative type assignment we define in Section 6. We will establish a relation between Classical Logic and the π-calculus through that notion of type assignment by showing that the translations also preserve types, *i.e.* the image of a typeable term gives a process that is a witness to the same judgement. We thereby establish that the π-calculus has a strong link to functional languages with control and is thereby inherently more expressive than just the λ-calculus; see also below where we discuss our results for Parigot's $\lambda\mu$-calculus [43].

Classical sequents

The *sequent calculus* LK, introduced by Gentzen in [28], 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 [28]) 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 $B_1\vee\ldots\vee B_m$. Kleene's version G_3 [39], with implicit weakening and contraction, of Implicative LK has four rules: *axiom*, *cut*, *left introduction* of the arrow, and *right introduction*:

$$(Ax) : \frac{}{\Gamma, A \vdash_{\rm LK} A, \Delta} \qquad (cut) : \frac{\Gamma \vdash_{\rm LK} A, \Delta \quad \Gamma, A \vdash_{\rm LK} \Delta}{\Gamma \vdash_{\rm LK} \Delta}$$

$$({\Rightarrow}R) : \frac{\Gamma, A \vdash_{\rm LK} B, \Delta}{\Gamma \vdash_{\rm LK} A{\Rightarrow}B, \Delta} \qquad ({\Rightarrow}L) : \frac{\Gamma \vdash_{\rm LK} A, \Delta \quad \Gamma, B \vdash_{\rm LK} \Delta}{\Gamma, A{\Rightarrow}B \vdash_{\rm LK} \Delta}$$

Since LK has no elimination 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, defined via local reductions of the proof-tree, which has –with some discrepancies– the flavour of term rewriting [40] or the evaluation of explicit substitutions [19, 1]. His *Hauptsatz* result expresses that this kind of proof reduction is normalising.

The calculus $\mathcal{LK}$[1] achieves a Curry-Howard isomorphism - first discovered for Combina-

[1] Since the main difference between $\mathcal{X}$ [8, 9] and $\mathcal{LK}$ is in the reduction rules, the observations in this section

tory Logic [27] - for the proofs in LK by constructing *witnesses* for derivable sequents. This is established by, similar to calculi like Parigot's $\lambda\mu$ and Curien and Herbelin's $\lambda\mu\tilde{\mu}$ [26], attaching Roman names to formulae in the left context, and Greek names to those on the right, and to associate syntactic structure 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 $\mathcal{LK}$: *sockets* (inputs, with Roman names) and *plugs* (outputs, with Greek names), that correspond to *variables* and *co-variables*, respectively, in [49], or to Parigot's λ and μ-variables (see also [26]).

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

Gentzen's proof reductions by *cut*-elimination[2] become the fundamental principle of computation in $\mathcal{LK}$. *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: *ergo*, a term is in normal form if and only if it has no sub-term of this shape. Note that reduction in $\mathcal{LK}$ is not confluent; 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.

Related work

Logic and computation

The relation between *logic* and *computation* hinges around the Curry-Howard isomorphism (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, thus linking the term's type to the proposition shown by the proof, and proof contractions become term reductions. This phenomenon was first discovered for Combinatory Logic [27], and played an important part in de Bruijn's Automath.[3]

Before Herbelin's PhD [33] and Urban's PhD [46], the study of the relation between computation, programming languages and logic has concentrated mainly on *natural deduction systems* (of course, exceptions exist [30, 31]). 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:

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

– 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 [29] and the Calculus of Constructions [24]. In fact, all these calculi are *applicative* in that abstraction and application (corresponding to arrow introduction and elimination) are the main constructors in the syntax.

are true also for $\mathcal{X}$.

[2] In his original paper [28], Gentzen never considered progressing a *cut* over a *cut*. In that sense, reduction in $\mathcal{LK}$ as we consider it here is much more 'liberal'; this comes at the price of losing strong normalisation of typeable terms.

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

The link between Classical Logic and continuations and control was first established by Griffin for the λ_C-Calculus [32] (where C stands for Felleisen's C operator). Not much later, Parigot presented his $\lambda\mu$-calculus [43], 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* ($\bot$):

$$(\bot):\ \frac{\Gamma \vdash_{\Lambda\mu} M : A \mid \alpha{:}A,\Delta}{\Gamma \vdash_{\Lambda\mu} [\alpha]M : \bot \mid \alpha{:}A,\Delta} \qquad (\mu):\ \frac{\Gamma \vdash_{\Lambda\mu} M : \bot \mid \alpha{:}A,\Delta}{\Gamma \vdash_{\Lambda\mu} \mu\alpha.M : A \mid \Delta}$$

It uses two disjoint sets of variables (Roman and Greek characters). That control can be expressed in the lazy variant of $\lambda\mu$ was shown in [25].

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 [45]). This imbalance can be observed in the $\lambda\mu$-calculus: adding $\bot$ as pseudo-type (only negation, or $A \rightarrow \bot$, is expressed; $\bot \rightarrow A$ is not a type), the $\lambda\mu$-calculus corresponds to *minimal classical logic* [5].

Herbelin has studied the calculus $\lambda\mu\tilde{\mu}$ as an extension of $\lambda\mu$ without *application*, which gives a fine-grained account of manipulation of sequents [33, 26, 34]. The relation between call-by-name and call-by-value in the fragment of LK with negation and conjunction is studied in Wadler's Dual Calculus [49]; as in calculi like $\lambda\mu$ and $\lambda\mu\tilde{\mu}$, 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 $\lambda\mu\tilde{\mu}$ has been investigated in [8, 9]; there it was shown that it is straightforward to map $\lambda\mu\tilde{\mu}$-terms into $\mathcal{X}$ whilst preserving reduction, but that it is only partially possible to do the converse.

π-calculus and logic

In the past, there have been several investigations of translations from various calculi (or logics) into the π-calculus [42], starting with Milner's seminal paper, presenting his input-based translation of the λ-calculus [14] into the π-calculus, and showing that the translation of closed λ-terms respects *lazy* reduction to normal form up to substitution. Many papers have been published in that area; here we concentrate on a review of the literature on the relationship between logic and the π-calculus.

The original idea of giving a computational translation of the *cut* as a communication primitive that we propose in this paper is also used by Abramsky in [4]; that paper was more a philosophical exposition of ideas, rather than a detailed presentation of an encoding with proofs. Abramsky's ideas were taken further by Bellin and Scott [17] and later by Bruscali and Gugliemi [21, 20]. On the relation between Girard's linear logic [30] and the π-calculus, Bellin and Scott [17] give a treatment of information flow in proof-nets; only a small fragment of Linear Logic was considered, and the translation between proofs and π-calculus was left rather implicit as also noted by [22].

To illustrate this, notice that [17] uses the standard syntax for the polyadic π-calculus

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

similar to the one we use here (see Definition 3.1) but for the fact that for us output is not synchronous, and there the *let*-construct is not used. However, the encoding of a 'cut' in linear logic

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

i.e. the 'term' $x{:}A \otimes B, m{:}A^{\perp}, w{:}B^{\perp}$, gets translated in [17] into a 'language of proofs' which looks like:

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

where the terms Cut and I are (rather loosely) defined. Notice the use of arbitrary application of processes to channel names, and the operation of pairing; the authors of [17] do not specify how to relate this notation to the above syntax of processes they consider.

However, even if this relationship is made explicit, even then a different π-calculus is needed to make the encoding work. To clarify this point, consider the translation in the π-calculus of the term above, which according to the definition given in [17] becomes:

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

Although intended, no communication is possible in this term (we have underlined the desired communication which is impossible, as the arity of the channel k does not match). To overcome this kind of problem, Bellin and Scott would need the *let*-construct with use of pairs of names as we have introduced in this paper in Definition 3.1. Moreover, there is no relation between the interpreted terms and proofs stated in [17] in terms of logic, types, or provable statements; here, we make a clear link between interpreted proofs and the logic through our notion of type assignment for the π-calculus.

Honda and Laurent [35] studied a typed π-calculus and show that a specific form of polarised linear logic [41] and a typed version of the asynchronous π-calculus [38] are essentially different ways of presenting the same structure. In contrast, our translations are very natural and intuitive by interpreting the *cut* operationally as a synchronisation in the basic, untyped π-calculus.

Honda, Yoshida, and Berger [38] study a relation between a typed (*i.e.* types are part of the syntax of a term) Call-by-Value $\lambda\mu$ and a linear π-calculus. The syntax of processes there considered is

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

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

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

as the basic synchronisation rule, as well as

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

where $\mathrm{C}[\cdot]$ is an arbitrary (typed) context; note that $\searrow$ synchronises with any occurrence of $\overline{x}\,\vec{a}$, no matter what guards they may be placed under. The resulting calculus is thereby very different from the original π-calculus. Types for processes prescribe usage of names, and name passing is restricted to *bound (private, hidden) name passing.*[4]

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

The translation of typed $\lambda\mu$ they present is type dependent, in that, for each term, there are different π-processes assigned, depending on the original type; this makes the translation quite cumbersome. That paper achieves a *full abstraction* result, but at the price of considering only an explicitly typed version of $\lambda\mu$, restricted to CBV, and then only the lazy version of that, since it is essentially based on Milner's translation, *i.e.* does not model reduction in the right-hand side of an application; expressiveness of the results is obtained by changing the reduction strategy of the π-calculus by allowing synchronisations also under guard and under replication. So the results of [38] and our paper cannot really be compared; we just remark that our translation is type-free, maps onto a version of the π-calculus that does not allow for reduction to take place under replication or guard, deals with untypeable terms as well, and that our semantic translation deals with reduction in a sequent calculus.

An accurate and elegant result on the relationship between π-calculus and linear logic is achieved by Beffara [15, 16]. In particular, in [16] various mappings of the $\lambda\mu$-calculus with linear logic types are encoded into synchronous π-calculus with forwarders. Observe that $\lambda\mu$ encodes a 'natural deduction' style of reasoning, while in this paper we are considering a sequent calculus kind of reasoning for classical logic.

In [12], two of the authors presented a compositional output-based translation for the $\Lambda\mu$-calculus (a variant of $\lambda\mu$ with separate naming and μ-binding operations) extended with explicit substitution, into the π-calculus with pairing, and showed that this translation preserves single-step explicit head reduction with respect to contextual equivalence. Since $\Lambda\mu$ is a λ-calculus where reduction is confluent, the result of [12] is only partial with respect to the results we present here. They showed a full abstraction result for their encoding in [13], but restricted to $\lambda\mu$.

A result on the relation between classical logic and the π-calculus has appeared as [23], but for the fact that there a relation is established between the $\lambda\mu\tilde{\mu}$-calculus and the π-calculus; since the focus for the translation as defined there is termination, it preserves only outermost reduction, which does not get formally motivated as a significant restriction of (proof)-reduction. 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.

Main results

In this paper, we will show results for two interpretations of $\mathcal{LK}$ into the π-calculus, each with their own strengths and provable properties. We will first present a natural translation $[\![\cdot]\!]_{\mathrm{N}}$ that respects a notion of head reduction through synchronisation, and a semantic translation $[\![\cdot]\!]_{\mathrm{S}}$ that respects weak bisimilarity in full. Although the origin of terms in $\mathcal{LK}$ are the proofs in LK, these translations in no way depend on type information, but map type-free terms (so also terms that do not correspond to proofs) to type-free processes. The translations focus, as is usual in semantics, on *observable behaviour*, and as mentioned above we will show that, if P reduces to Q, then the observable behaviour of Q is included in that of P, and that individual reduction steps are preserved in that sense.

The first translation is called *natural* since it closely follows the nature and structure of proofs in LK, and is, in approach, closely related to Milner's encoding and the output-based spine translation of [11]. The results we will show for this translation are:

Soundness (Theorems 4.8 and 7.8): If P reduces to Q using *head* reduction, then the observational behaviour of $[\![P]\!]_{\mathrm{N}}$ contains that of $[\![Q]\!]_{\mathrm{N}}$ (as we will see in Remark 2.7, given the non-confluent nature of both $\mathcal{LK}$ and the π-calculus, we cannot show that $[\![P]\!]_{\mathrm{N}} = [\![Q]\!]_{\mathrm{N}}$)

and if all head-reduction paths from P contain an infinite number of $(exp\text{-}imp)$ steps, then $[\![P]\!]_{\text{N}}$ diverges.

Preservation of types (Theorems 6.8 and 7.10): If P is a witness for the judgement $\Gamma \vdash \Delta$, then so is $[\![P]\!]_{\text{N}}$, effectively showing that all proofs in LK have a representation in the π-calculus.

These results show that the natural translation is strong, but as translation of full *cut*-elimination it falls short: not all reductions are modelled, and the translation is not *complete* (see Example 4.12). The first is almost standard in the literature: for example, [42, 44] can only model *lazy* reduction, and, as argued in [11], only *explicit lazy* reduction in a step-by-step fashion. The second is a direct consequence of the fact that reduction in $\mathcal{LK}$ is a term rewriting system, where pattern matching takes place; we cannot fully express in the π-calculus. The only way to 'hold' the reduction of the synchronisation in the implementation of the rule $(\dagger imp\text{-}out)$ (as in Example 4.12) would be to put the whole process under input, thereby blocking other *cut*s as well. That the process involved in the synchronisation is, in fact, the interpretation of an *import* can not be seen from the process itself.

We will show here that we can overcome these restrictions by presenting a second translation, $[\![\cdot]\!]_{\text{S}}$, that we call *semantical*; it interprets terms as infinite resources, and is capable of representing *cut*-elimination in full, albeit not through mimicking reduction, but through bisimilarity (hence the moniker "semantical"). It is a generalisation of the natural translation in that it treats the interaction between a term and a context not through an input over the translation of the latter, as the natural translation does. For this second translation, we will show:

Operational Soundness (Theorems 5.4 and 7.8): If P reduces to Q in $\mathcal{LK}$'s full reduction, then the observational behaviour of $[\![P]\!]_{\text{S}}$ contains that of $[\![Q]\!]_{\text{S}}$.

Preservation of types (Theorems 6.9 and 7.11): If P is a witness for the judgement $\Gamma \vdash \Delta$, so is $[\![P]\!]_{\text{S}}$.

Operational completeness (Theorem 5.6): If $[\![P]\!]_{\text{S}}$ can be executed, then so can P, and these executions are related via $[\![\cdot]\!]_{\text{S}}$ by: if $[\![P]\!]_{\text{S}} \rightarrow_{\pi} \mathsf{Q}$ then there exists $P' \in \mathcal{LK}$, R such that $\mathsf{Q} \rightarrow_{\pi}^{*} \mathsf{R}$, $!\,\mathsf{R} \approx [\![P']\!]_{\text{S}}$, and $P \rightarrow^{*} P'$.

Preservation of typeable termination (Corollary 5.10): If P is typeable, then $[\![P]\!]_{\text{S}}$ is bisimilar to a process in normal form; we hereby emulate Gentzen's *Hauptsatz* result; this is not possible for the normal encoding.

There are many more properties that one could demand to hold for these translations, like preservation of *compositions*, of *termination*, of *simulations*, of *equivalences*, *full abstraction*, etc. It is not immediately clear if checking these properties, or even aiming for them, makes sense in the context of the translations of $\mathcal{LK}$ we define here. After all, we are not interpreting one model of computation into another, but rather study the relation between cut-elimination in classical logic and communication in a process calculus. The calculus $\mathcal{LK}$ we present here has not been proposed as a calculus to represent computation, is not a programming language and should not be treated as such; so it seems unreasonable to demand that the criteria we set on models of computation should also hold for $\mathcal{LK}$.

We will see that, for the kind of cut-elimination for LK as we consider in this paper, when allowing *cut*-over-*cut* reduction, *cut*-elimination is highly non-terminating, even looping for terms that intuitively should not; since for many terms that have a finite reduction path also a looping reduction path exists and our translations respect single reduction steps, we cannot hope to show that termination is preserved by our translations. However, this does not imply that *no* termination results can be shown; in fact, for the semantic translation we will show that Gentzen's *Hauptsatz* result (*i.e.* every provable judgement $\Gamma \vdash \Delta$ has a *cut*-free proof) is, in a sense, preserved.

In [7], we first presented our results on the translation of $\mathcal{LK}$-terms into the π-calculus; that paper also presented the notion of type assignment as defined here, as well as a proof that type assignment and cut-elimination are preserved by the translation. Since some details of the translations differ, we repeat these results here, with all particulars of the proofs; moreover, here we define head reduction '$\rightarrow_{\text{H}}$' for $\mathcal{LK}$, and show that the translation $[\![\cdot]\!]_{\text{N}}$ respects $\rightarrow_{\text{H}}$; we also add the semantic translation $[\![\cdot]\!]_{\text{S}}$ and show that this is faithful with respect to $\mathcal{LK}$'s full reduction. In addition to [7] (and [9]), we treat the connective $\neg$ as well.

Overview of this paper

In Section 1, we give the definition of (implicative) $\mathcal{LK}$, followed by the notion of type assignment which establishes the Curry-Howard isomorphism. In Section 2, we show how to rewrite $\mathcal{LK}$-terms, and show the relation with LK's *cut*-elimination. The π-calculus with pairing is presented in Section 3. Section 4 defines the natural translation $[\![\cdot]\!]_{\text{N}}$ of $\mathcal{LK}$-terms into π-processes that closely follows the intuition of $\mathcal{LK}$, and shows a soundness and type-preservation result. In Section 5 we will modify the natural translation to represent $\mathcal{LK}$'s reduction in *full*, via the semantic translation $[\![\cdot]\!]_{\text{S}}$ and show soundness, type-preservation, and completeness. We will use this translation to show that every typeable term corresponds to a process in normal form, which emulates Gentzen's *Hauptsatz* result. In Section 6, we define a notion of type assignment for the π-calculus, and show that, under the two interpretations, typeable terms translate to processes that are typeable in the same way. Then, in Section 7 we look at how to represent negation in $\mathcal{LK}$, and study the relation between that representation and reduction. We conclude by representing negation directly in π, and show that type assignment is preserved also here.

1 The calculus $\mathcal{LK}$

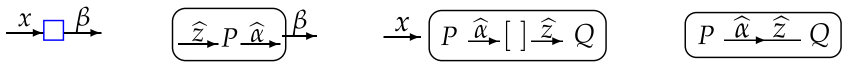
In this section and the next we will give the definition of $\mathcal{LK}$, a variant of the calculus $\mathcal{X}$ which has been proven to be a fine-grained implementation model for various well-known calculi [8], like the λ-calculus, $\lambda\mu$, and $\lambda\mu\tilde{\mu}$. As discussed in the introduction, the calculus $\mathcal{LK}$ is linked to Gentzen's sequent calculus LK; the system we will consider in this section has only implication, no structural rules and a changed axiom. $\mathcal{LK}$ features two separate categories of 'connectors', *plugs* and *sockets*, that act as output and input channels, respectively, and is defined without any notion of substitution or application. For the sake of clarity, we will develop our results first just for the implicative fragment of LK; we will consider negation in Section 7.

We would like to stress that the calculus $\mathcal{LK}$ we define here is not proposed as an abstract machine to model computation ($\mathcal{X}$ and $\lambda\mu\tilde{\mu}$ are better suited for that), but just as a language that allows us to treat Gentzen's *cut*-elimination in a syntactical manner; we will see that reduction is highly inefficient, since looping on normalisable terms.

Definition 1.1 (SYNTAX) The terms of the $\mathcal{LK}$-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*.

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

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 other calculi:

$\mathcal{LK}$	$\lambda\mathbf{x}$	$\Lambda\mu$	$\lambda\mu\tilde{\mu}$
$\langle x{\cdot}\beta\rangle$	x	$[\beta]x$	$\langle x\,\vert\,\beta\rangle$
$\widehat{z}P\widehat{\alpha}{\cdot}\beta$	$\lambda z.P$	$[\beta]\mu\alpha.\lambda z.P$	$\langle\lambda z.\mu\alpha.P\,\vert\,\beta\rangle$
$P\widehat{\alpha}\,[x]\,\widehat{z}Q$	$xPQ_1\cdots Q_n$	$[\omega](\mu\alpha.P)(\lambda v.[\omega]xv\lambda z.Q)$	$\langle x\,\vert\,\mu\alpha.P\cdot\tilde{\mu}z.Q\rangle$
$P\widehat{\alpha}\dagger\widehat{z}Q$	$Q\,\langle z{:=}P\rangle$	$[\omega](\mu\alpha.P)(\lambda z.Q)$	$\langle\mu\alpha.P\,\vert\,\tilde{\mu}z.Q\rangle$

(where $\lambda\mathbf{x}$ is Bloo and Rose's λ-calculus with explicit substitution [18], and in the third case Q is seen as a context, acting as a stack of terms $Q_1,\ldots,Q_n$; for details, see [6] and [9]).

The encoding of the λ-calculus, $\lambda\mathbf{x}$, and $\lambda\mu$ into $\mathcal{LK}$ are defined in [9] through:

$$\begin{array}{rcl} \ulcorner x\lrcorner_{\alpha} & \triangleq & \langle x{\cdot}\alpha\rangle \\ \ulcorner\lambda x.M\lrcorner_{\alpha} & \triangleq & \widehat{x}\ulcorner M\lrcorner_{\beta}\widehat{\beta}{\cdot}\alpha \\ \ulcorner MN\lrcorner_{\alpha} & \triangleq & \ulcorner M\lrcorner_{\gamma}\widehat{\gamma}\dagger\widehat{x}(\ulcorner N\lrcorner_{\beta}\widehat{\beta}\,[x]\,\widehat{y}\langle y{\cdot}\alpha\rangle) \\ \ulcorner M\,\langle x{:=}N\rangle\lrcorner_{\alpha} & \triangleq & \ulcorner N\lrcorner_{\beta}\widehat{\beta}\dagger\widehat{x}\ulcorner M\lrcorner_{\alpha} \\ \ulcorner\mu\delta.[\gamma]M\lrcorner_{\alpha} & \triangleq & \ulcorner M\lrcorner_{\gamma}\widehat{\delta}\dagger\widehat{x}\langle x{\cdot}\alpha\rangle \end{array}$$

Notice that terms are defined 'under output'. That paper also defines an interpretation of $\lambda\mu\tilde{\mu}$ into $\mathcal{LK}$:

$$\ulcorner\langle v\,\vert\,e\rangle\lrcorner \;\triangleq\; \ulcorner v\lrcorner_{\alpha}\widehat{\alpha}\dagger\widehat{x}\ulcorner e\lrcorner_{x}$$

$$\begin{array}{rcl@{\qquad}rcl} \ulcorner x\lrcorner_{\alpha} & \triangleq & \langle x{\cdot}\alpha\rangle & \ulcorner\alpha\lrcorner_{x} & \triangleq & \langle x{\cdot}\alpha\rangle \\ \ulcorner\lambda x.v\lrcorner_{\alpha} & \triangleq & \widehat{x}\ulcorner v\lrcorner_{\beta}\widehat{\beta}{\cdot}\alpha & \ulcorner v{\cdot}e\lrcorner_{x} & \triangleq & \ulcorner v\lrcorner_{\alpha}\widehat{\alpha}\,[x]\,\widehat{y}\ulcorner e\lrcorner_{y} \\ \ulcorner\mu\beta.c\lrcorner_{\alpha} & \triangleq & \ulcorner c\lrcorner\widehat{\beta}\dagger\widehat{x}\langle x{\cdot}\alpha\rangle & \ulcorner\tilde{\mu}y.c\lrcorner_{x} & \triangleq & \langle x{\cdot}\beta\rangle\,\widehat{\beta}\dagger\widehat{y}\ulcorner c\lrcorner \end{array}$$

Here terms are interpreted under output, and contexts under input.

Definition 1.2 *i*) The *bound sockets* and *bound plugs* in a term are defined by:

$$\begin{array}{lcl} bs(\langle x{\cdot}\alpha\rangle) & = & \emptyset \\ bs(\widehat{z}P\widehat{\alpha}{\cdot}\beta) & = & bs(P)\cup\{z\} \\ bs(P\widehat{\alpha}\,[x]\,\widehat{z}Q) & = & bs(P)\cup bs(Q)\cup\{z\} \\ bs(P\widehat{\alpha}\dagger\widehat{z}Q) & = & bs(P)\cup bs(Q)\cup\{z\} \end{array}$$

$$\begin{array}{lcl} bp(\langle x{\cdot}\alpha\rangle) & = & \emptyset \\ bp(\widehat{z}P\widehat{\alpha}{\cdot}\beta) & = & bp(P)\cup\{\alpha\} \\ bp(P\widehat{\alpha}\,[x]\,\widehat{z}Q) & = & bp(P)\cup\{\alpha\}\cup bp(Q) \\ bp(P\widehat{\alpha}\dagger\widehat{z}Q) & = & bp(P)\cup\{\alpha\}\cup bp(Q) \end{array}$$

ii) The set of *bound connectors* of P is defined by: $bc(P) = bs(P)\cup bp(P)$.

iii) A socket x or plug α occurring in P which is not bound is called *free*, written $x \in fs(P)$ and $\alpha \in fp(P)$.

We will identify terms that only differ in the names of bound connectors, as usual, and write $x \notin fs(P,Q)$ for $x \notin fs(P)$ & $x \notin fs(Q)$, etc.

Notice that each term in $\mathcal{LK}$ has at least *one* free plug.

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.

In order to come to a notion of type (or better: context) assignment for $\mathcal{LK}$, we define types and contexts.

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

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

where φ is a basic type of which there are countably many.[5]

ii) 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 $x{:}A_1 \in \Gamma_1$ and $x{:}A_2 \in \Gamma_2$ then $A_1 = A_2$), and write $\Gamma,x{:}A$ for $\Gamma,\{x{:}A\}$. We write $x \in \Gamma$ if there exists A such that $x{:}A \in \Gamma$, and $x \notin \Gamma$ if this is not the case. We write $\Gamma\backslash x$ for $\Gamma \setminus \{x{:}A\}$ if $x \in \Gamma$ or Γ if $x \notin \Gamma$.

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

So, when writing a context as $\Gamma,x{:}A$, this implies that $x{:}A \in \Gamma$, or Γ is not defined on x.

The notion of type assignment on $\mathcal{LK}$ that we present in this section is Kleene's 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.

Definition 1.4 (Typing for $\mathcal{LK}$) *i*) *Type judgements*[6] for $\mathcal{LK}$ are expressed via a ternary relation $P : \Gamma \vdash_{\text{LK}} \Delta$, 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.

ii) *Type assignment for* $\mathcal{LK}$ is defined by the following rules:

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

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

We write $P : \Gamma \vdash_{\text{LK}} \Delta$ if there exists a derivation using these rules that has this judgement in the bottom line.

It is easy to show that weakening is admissible.

Notice that each term in $\mathcal{LK}$ has at least *one* free plug, so it is impossible to derive a statement like $P : \Gamma \vdash_{\text{LK}} \emptyset$[7] and 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.

[5] These types are normally known as *natural* (or *Curry*) types.

[6] We use the notation of [9].

[7] This is possible in the extended system we will consider in Section 7.

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

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

and its inhabitation in $\mathcal{LK}$:

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

2 Reduction on $\mathcal{LK}$

The reduction rules for the calculus $\mathcal{LK}$ are directly inspired by the *cut*-elimination rules in lk. It is possible to define proof reduction in many ways; Gentzen decided to consider the simplest contractions, and considered only the last rule applied in the two sub-derivations of *cut*s:

$$\dfrac{\dfrac{\boxed{\phantom{\mathcal{D}}}}{\Gamma \vdash_{\text{LK}} A,\Delta}\,(r) \qquad \dfrac{\boxed{\phantom{\mathcal{D}}}}{\Gamma,A \vdash_{\text{LK}} \Delta}\,(l)}{\Gamma \vdash_{\text{LK}} \Delta}\,(cut)$$

In case the formula A is introduced in both these sub-derivations (*i.e.* either $({\Rightarrow}R)$ and $({\Rightarrow}L)$, or (Ax) and $({\Rightarrow}L)$, or $({\Rightarrow}R)$ and (Ax), or (Ax) and (Ax)) the *cut* can be contracted directly; otherwise, a sub-proof gets 'pushed' into the one does not introduce the formula, one proof-step at the time; notice that this might apply to both, so a choice might have to be made, which in itself might lead to different results, *i.e.* different proofs for the same sequent.

We model these proof-contraction steps via term rewriting rules for $\mathcal{LK}$. For example, since

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

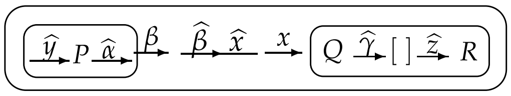
contracts to both:

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

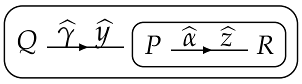
and

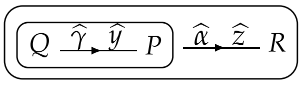
$$\dfrac{\begin{array}{c}\boxed{\mathcal{D}_2}\\ \Gamma \vdash_{\mathbf{LK}} A,\Delta\end{array} \qquad \dfrac{\begin{array}{c}\boxed{\mathcal{D}_1}\\ \Gamma,A \vdash_{\mathbf{LK}} B,\Delta\end{array} \qquad \dfrac{\begin{array}{c}\boxed{\mathcal{D}_3}\\ \Gamma,B \vdash_{\mathbf{LK}} \Delta\end{array}}{\Gamma,A,B \vdash_{\mathbf{LK}} \Delta}\,(Wk)}{\Gamma,A \vdash_{\mathbf{LK}} \Delta}\,(cut)}{\Gamma \vdash_{\mathbf{LK}} \Delta}\,(cut)$$

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



reduces to both $Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\alpha} \dagger \widehat{z}R)$ and $(Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\alpha} \dagger \widehat{z}R$





being the witnesses for the two resulting proofs; also this might lead to different results (*i.e. cut*-free proofs).

This behaviour is reflected in rule (*exp-imp*), as presented in Definition 2.2. We can see the *cut* $(\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R)$ as a function $\widehat{y}P\widehat{\beta}\cdot\alpha$ (with body P, that takes input on y and outputs on β) 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 R the context of the 'explicit substitution' of P in Q). 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.[8]

Following Gentzen's approach, $\mathcal{LK}$'s term rewriting rules explain 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 nodes to points in the term where they can interact. For this, it is important to know when a connector is introduced, *i.e.* is exposed and unique; informally, a term P introduces a socket x if P contains x and 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 α*: Either $P = \widehat{x}Q\widehat{\beta}\cdot\alpha$ and $\alpha \notin fp(Q)$, or $P = \langle x\cdot\alpha\rangle$.
P introduces x: Either $P = Q\widehat{\beta}\,[x]\,\widehat{y}R$ with $x \notin fs(Q,R)$, or $P = \langle x\cdot\alpha\rangle$.

The logical 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.

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

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

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 have the same normal forms (since reduction is not confluent, 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. In case both connectors are not introduced, this search can start in either direction, giving another source of non-confluence.

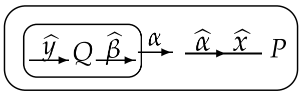
Similarly to the reasoning above, also the rules dealing with propagating *cuts* are inspired by Gentzen's *cut*-elimination rules. Take

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

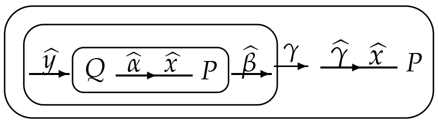
(notice the contraction towards $A{\Rightarrow}B$ in the left-hand sub-derivation, so the plug associated to this formula would not be introduced in the witness for $\Gamma \vdash_{\mathrm{LK}} A{\Rightarrow}B,\Delta$) which reduces to

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

Notice that now in the conclusion of the left-hand sub-derivation, the formula $A{\Rightarrow}B$ is not contracted: therefore, in the witness for this proof, this is represented by an introduced plug; in fact, the witness for the first proof, the term $(\widehat{y}Q\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}P$, reduces to the witness for the second proof $(\widehat{y}(Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{x}P$ where now γ is introduced,[9] as reflected in rule (*exp-out*†) below. So the diagram



with α free in Q, reduces to:



Also, since

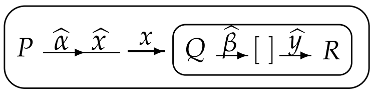
[9] We rename the outermost α to γ in order to adhere to Barendregt's convention.

$$\dfrac{\begin{array}{c}\boxed{\mathcal{D}_1}\\ \Gamma \vdash_{\mathrm{LK}} A{\Rightarrow}B,\Delta\end{array} \qquad \dfrac{\begin{array}{c}\boxed{\mathcal{D}_2}\\ \Gamma,A{\Rightarrow}B \vdash_{\mathrm{LK}} A,\Delta\end{array} \quad \begin{array}{c}\boxed{\mathcal{D}_3}\\ \Gamma,A{\Rightarrow}B,B \vdash_{\mathrm{LK}} \Delta\end{array}}{\Gamma,A{\Rightarrow}B \vdash_{\mathrm{LK}} \Delta}\,({\Rightarrow}L)}{\Gamma \vdash_{\mathrm{LK}} \Delta}\,(cut)$$

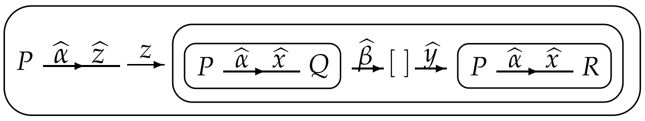
(again, notice the contraction) reduces to

$$\dfrac{\begin{array}{c}\boxed{\mathcal{D}_1}\\ \Gamma \vdash_{\mathrm{LK}} A{\Rightarrow}B,\Delta\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}\boxed{\mathcal{D}_1}\\ \Gamma \vdash_{\mathrm{LK}} A{\Rightarrow}B,\Delta\end{array} \quad \begin{array}{c}\boxed{\mathcal{D}_2}\\ \Gamma,A{\Rightarrow}B \vdash_{\mathrm{LK}} A,\Delta\end{array}}{\Gamma \vdash_{\mathrm{LK}} A,\Delta}\,(cut) \quad \dfrac{\begin{array}{c}\boxed{\mathcal{D}_1}\\ \Gamma \vdash_{\mathrm{LK}} A{\Rightarrow}B,\Delta\end{array} \quad \begin{array}{c}\boxed{\mathcal{D}_3}\\ \Gamma,A{\Rightarrow}B,B \vdash_{\mathrm{LK}} \Delta\end{array}}{\Gamma,B \vdash_{\mathrm{LK}} \Delta}\,(cut)}{\Gamma,A{\Rightarrow}B \vdash_{\mathrm{LK}} \Delta}\,({\Rightarrow}L)}{\Gamma \vdash_{\mathrm{LK}} \Delta}\,(cut)$$

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



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



as reflected in rule $(\dagger\mathit{imp\text{-}out})$.

This leads to the next set of rules that deal with cuts that do not have both connectors introduced, and define how to move that *cut* inwards.

Definition 2.3 (Propagation rules) Left propagation:

$$\begin{array}{rrcll}
(cap\dagger): & \langle y{\cdot}\beta\rangle\widehat{\alpha} \dagger \widehat{x}P & \to & \langle y{\cdot}\beta\rangle & (\beta \neq \alpha)\\
(exp\text{-}out\dagger): & (\widehat{y}Q\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}P & \to & (\widehat{y}(Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{x}P & \\
 & & & & (\gamma\ \mathit{fresh}, \alpha\ \mathit{not\ introduced})\\
(exp\text{-}in\dagger): & (\widehat{y}Q\widehat{\beta}{\cdot}\gamma)\widehat{\alpha} \dagger \widehat{x}P & \to & \widehat{y}(Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma & (\gamma \neq \alpha)\\
(imp\dagger): & (Q\widehat{\beta}\,[z]\,\widehat{y}R)\widehat{\alpha} \dagger \widehat{x}P & \to & (Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta}\,[z]\,\widehat{y}(R\widehat{\alpha} \dagger \widehat{x}P) & \\
(cut\dagger): & (Q\widehat{\beta} \dagger \widehat{y}R)\widehat{\alpha} \dagger \widehat{x}P & \to & (Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta} \dagger \widehat{y}(R\widehat{\alpha} \dagger \widehat{x}P) &
\end{array}$$

Right propagation:

$$\begin{array}{rlcll}
(\dagger cap): & P\widehat{\alpha} \dagger \widehat{x}\langle y{\cdot}\beta\rangle & \to & \langle y{\cdot}\beta\rangle & (y \neq x)\\
(\dagger exp): & P\widehat{\alpha} \dagger \widehat{x}(\widehat{y}Q\widehat{\beta}{\cdot}\gamma) & \to & \widehat{y}(P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta}{\cdot}\gamma & \\
(\dagger imp\text{-}out): & P\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R) & \to & P\widehat{\alpha} \dagger \widehat{z}((P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \dagger \widehat{x}R)), & \\
 & & & & (z\ \mathit{fresh}, x\ \mathit{not\ introduced})\\
(\dagger imp\text{-}in): & P\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[z]\,\widehat{y}R) & \to & (P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \dagger \widehat{x}R) & (z \neq x)\\
(\dagger cut): & P\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R) & \to & (P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \dagger \widehat{x}R) &
\end{array}$$

Definition 2.4 (Reduction) *i*) We write $\to^{*}_{\mathcal{LK}}$ for the reduction relation defined as the smallest pre-order that includes the logical and propagation rules, extended with the contextual rules[10]

[10] Reduction in $\mathcal{LK}$ is defined as a term rewriting system, where the contextual rules are normally left implicit; we mention them here because we define a restriction of reduction that also limits the contextual rules.

$$\begin{array}{lll}
(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma}\,\boldsymbol{\dagger}\,\widehat{u}(Q\widehat{\tau}\,[u]\,\widehat{x}R) & \rightarrow_{\mathcal{LK}} & (\dagger\textit{imp-out}) \\
(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma}\dagger\widehat{y}(((\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma}\dagger\widehat{u}Q)\widehat{\tau}\,[y]\,\widehat{x}((\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma}\,\boldsymbol{\dagger}\,\widehat{u}R)) & \rightarrow_{\mathcal{LK}} & (\dagger\textit{cap}) \\
(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma}\dagger\widehat{y}(((\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma}\,\boldsymbol{\dagger}\,\widehat{u}Q)\widehat{\tau}\,[y]\,\widehat{x}R) & \rightarrow_{\mathcal{LK}} & (\textit{exp}),=_\alpha \\
(\widehat{v}\langle v\cdot\rho\rangle\widehat{\rho}\cdot\gamma)\widehat{\gamma}\dagger\widehat{y}((\widehat{z}P\widehat{\delta}\cdot\tau)\widehat{\tau}\,[y]\,\widehat{x}R) & \rightarrow_{\mathcal{LK}} & (\textit{exp-imp}) \\
(\widehat{z}P\widehat{\delta}\cdot\tau)\widehat{\tau}\,\boldsymbol{\dagger}\,\widehat{v}(\langle v\cdot\rho\rangle\widehat{\rho}\dagger\widehat{x}R) & \rightarrow_{\mathcal{LK}} & (\dagger\textit{cut}) \\
((\widehat{z}P\widehat{\delta}\cdot\tau)\widehat{\tau}\,\boldsymbol{\dagger}\,\widehat{v}\langle v\cdot\rho\rangle)\widehat{\rho}\dagger\widehat{x}((\widehat{z}P\widehat{\delta}\cdot\tau)\widehat{\tau}\,\boldsymbol{\dagger}\,\widehat{v}R) & \rightarrow_{\mathcal{LK}} & (\textit{exp},\dagger\textit{cap}) \\
(\widehat{z}P\widehat{\delta}\cdot\rho)\widehat{\rho}\,\boldsymbol{\dagger}\,\widehat{x}R & \rightarrow_{\mathcal{LK}} & (\textit{exp}) \qquad \widehat{z}P\widehat{\delta}\cdot\sigma
\end{array}$$

Figure 1: Running $(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma}\dagger\widehat{u}(Q\widehat{\tau}\,[u]\,\widehat{x}R)$ of Example 2.5.

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

ii) We define the notion of *head* reduction, $\rightarrow_{\mathrm{H}}$, by excluding reductions in and toward *import*, via the *elimination* of the propagation rules that move into an *import* (*i.e.* (*imp*$\dagger$), ($\dagger$*imp-out*), and ($\dagger$*imp-in*), as well as the second and third contextual rule).

iii) We define *innermost* reduction $\rightarrow_{\mathrm{I}}$ by allowing the rules to be applied only to *cut*s composed out of terms in normal form (*i.e.* that contain no *cut*s).

iv) We write $P\uparrow$ (and say that P *diverges*) if all reduction paths starting from P contain an infinite number of (*exp-imp*) steps.[11]

Notice that this notion of reduction has many critical pairs, making reduction highly non-confluent.

The main difference between this reduction and that of $\mathcal{X}$ is that, essentially, in $\mathcal{X}$ *activated cut*s $P\widehat{\alpha}\nearrow\widehat{x}Q$ and $P\widehat{\alpha}\nwarrow\widehat{x}Q$ are added to the syntax, and only those are allowed to propagate over non-activated *cut*s; this is crucial for the Strong Normalisation result as shown by Urban (for a syntactic variant of $\mathcal{X}$), without sacrificing expressivity [46]. The idea is that, once activated, a *cut* has to run to completion, and cannot be 'crossed' with another *cut*. Instead, the rewriting we consider here corresponds more closely to *free cut*-elimination.

Example 2.5 Taking $P = \langle z\cdot\delta\rangle$, $Q = \langle u\cdot\tau\rangle$ and $R = \langle x\cdot\sigma\rangle$ (notice that then u is not introduced in $Q\widehat{\tau}\,[u]\,\widehat{x}R$), we can reduce $(\widehat{z}P\widehat{\delta}\cdot\gamma)\widehat{\gamma}\dagger\widehat{u}(Q\widehat{\tau}\,[u]\,\widehat{x}R)$ as in Figure 1 (notice that we have marked the cut that gets contracted).

Unlike a similar notion for the λ-calculus, our notion of head reduction is not deterministic: notice that

$$(\widehat{y}((\widehat{v}P\widehat{\delta}\cdot\sigma)\widehat{\sigma}\dagger\widehat{z}\langle z\cdot\gamma\rangle)\widehat{\gamma}\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}\langle x\cdot\beta\rangle \;\rightarrow_{\mathrm{H}}\; \begin{cases}
(\widehat{y}(\widehat{v}P\widehat{\delta}\cdot\gamma)\widehat{\gamma}\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}\langle x\cdot\beta\rangle \\
\widehat{y}((\widehat{v}P\widehat{\delta}\cdot\sigma)\widehat{\sigma}\dagger\widehat{z}\langle z\cdot\gamma\rangle)\widehat{\gamma}\cdot\beta
\end{cases}$$

so both cuts can be contracted under $\rightarrow_{\mathrm{H}}$.

Notice that our *cut*-elimination is different from Gentzen's original (implicit) definition: he in fact did not consider a *cut*-over-*cut* step, and used *innermost* reduction for his *Hauptsatz* result (see Proposition 5.7).

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

[11] By seeing only *exp-imp* as a true computational step, all other rules are considered *administrative*, comparable to dealing with explicit substitution.

Theorem 2.6 (Witness Reduction for $\mathcal{LK}$) *If $P : \Gamma \vdash \Delta$, and $P \rightarrow^*_{\mathcal{LK}} Q$, then $Q : \Gamma \vdash \Delta$.*

Proof: As in [9]. $\square$

Although the reduction rules of $\mathcal{LK}$ are different from those of $\mathcal{X}$, since activated *cut*s are no longer used, given that activated *cut*s are typed in the same way as normal *cut*s the proof is almost identical to that presented in [9].

Remark 2.7 (On non-confluence) We have already remarked that the reduction relation is not confluent. In fact, let P and Q be such that α is not free in P and x is not free in Q, then we can show both $P\widehat{\alpha} \dagger \widehat{x}Q \rightarrow^*_{\mathcal{LK}} P$ and $P\widehat{\alpha} \dagger \widehat{x}Q \rightarrow^*_{\mathcal{LK}} Q$. So, in particular, a term P can have more than one normal form.

Now when interpreting a term through its set of normal forms via $\ulcorner\cdot\rfloor_{\mathbf{NF}}$, it is easy to show that, if $P \rightarrow^*_{\mathcal{LK}} Q$, then $\ulcorner Q\rfloor_{\mathbf{NF}} \subseteq \ulcorner P\rfloor_{\mathbf{NF}}$; 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 translations into the π-calculus: if $P \rightarrow^*_{\mathcal{LK}} Q$, then $[\![P]\!]$ has more observable behaviour than $[\![Q]\!]$, expressed via $[\![P]\!] \sqsupseteq_{\approx} [\![Q]\!]$; see below.

In [9, 10] some basic properties are shown for $\mathcal{X}$, 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 are valid also for $\mathcal{LK}$, and motivate the formulation of admissible rules:

Lemma 2.8 (Garbage Collection and Renaming [10])

$$\begin{array}{llllllll}
(\dagger gc): & P\widehat{\alpha} \dagger \widehat{x}Q & \rightarrow_{\mathcal{LK}} & Q & \textit{if } x \notin fs(Q) & (\dagger ren): & \langle z{\cdot}\alpha\rangle\widehat{\alpha} \dagger \widehat{x}P & \rightarrow_{\mathcal{LK}} & P[z/x] \\
(gc\dagger): & P\widehat{\alpha} \dagger \widehat{x}Q & \rightarrow_{\mathcal{LK}} & P & \textit{if } \alpha \notin fp(P) & (ren\dagger): & P\widehat{\beta} \dagger \widehat{z}\langle z{\cdot}\alpha\rangle & \rightarrow_{\mathcal{LK}} & P[\alpha/\beta]
\end{array}$$

We can have looping reductions, even without rule (*exp-imp*).

Example 2.9 As an example of a looping reduction, take:

$$\begin{array}{lll}
P\widehat{\alpha} \,\pmb{\dagger}\, \widehat{x}(\langle x{\cdot}\beta\rangle\,\widehat{\beta} \dagger \widehat{z}Q) & \rightarrow_{\mathcal{LK}} & (\dagger cut) \\
(P\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\beta\rangle)\widehat{\beta} \dagger \widehat{z}(P\widehat{\alpha} \,\pmb{\dagger}\, \widehat{x}Q) & \rightarrow_{\mathcal{LK}} & (\dagger gc) \quad (x \notin fs(Q)) \\
(P\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\beta\rangle)\widehat{\beta} \,\pmb{\dagger}\, \widehat{z}Q & \rightarrow_{\mathcal{LK}} & (cut\dagger) \\
(P\widehat{\beta} \,\pmb{\dagger}\, \widehat{z}Q)\widehat{\alpha} \dagger \widehat{x}(\langle x{\cdot}\beta\rangle\,\widehat{\beta} \dagger \widehat{z}Q) & \rightarrow_{\mathcal{LK}} & (gc\dagger) \quad (\beta \notin fp(P)) \\
P\widehat{\alpha} \dagger \widehat{x}(\langle x{\cdot}\beta\rangle\,\widehat{\beta} \dagger \widehat{z}Q) & &
\end{array}$$

Moreover, assuming $P : \Gamma \vdash \alpha{:}A,\Delta$ and $Q : \Gamma,z{:}A \vdash \Delta$, we can construct a derivation for $P\widehat{\alpha} \dagger \widehat{x}(\langle x{\cdot}\beta\rangle\,\widehat{\beta} \dagger \widehat{z}Q) : \Gamma \vdash \Delta$ and all the intermediate terms in the reduction above are typeable by the Witness Reduction result: so also *cut*-elimination does not terminate and typeability does not guarantee termination.

Notice that, following the innermost reduction path, this loop is immediately broken:

$$P\widehat{\alpha} \dagger \widehat{x}(\langle x{\cdot}\beta\rangle\,\widehat{\beta} \,\pmb{\dagger}\, \widehat{z}Q) \;\rightarrow_{\mathcal{LK}} (\dagger ren)\;\; P\widehat{\alpha} \dagger \widehat{x}Q[x/z] \;=_{\alpha}\; P\widehat{\alpha} \dagger \widehat{z}Q$$

We hazard a guess that this is why Gentzen considers only innermost reduction.

3 The asynchronous π-calculus with pairing

The notion of asynchronous π-calculus that we consider in this paper is different from the standard system defined by Honda and Tokoro in [36]. One reason for the change we make lies directly in the calculus that is going to be interpreted, $\mathcal{LK}$, in which a term can be constructed binding *two connectors simultaneously*. We will model function and context interaction into processes communication by sending *data* over channels, *i.e.* not just names, but also pairs of names, so, inspired by [2], add pairing: we introduce a structure over names, such that a channel may pass along not only names but also pairs of names (but not a pair of pairs). This does not imply that the calculus we consider is polyadic, however: always only *one* item can be sent, which is either a name of a pair of names. We also introduce the *let*-construct to deal with inputs of pairs of names that get distributed over the continuation.

To ease this definition, we deviate slightly from the normal practice, and write either Greek characters $\alpha,\beta,\upsilon,\ldots$ or Roman characters $x,y,z,\ldots$ for channel names; we use a,b,c,n for either a Greek or a Roman name.

The reason we use the asynchronous π-calculus rather than the normal synchronous variant will become clear after Definition 4.1; notice that this choice is not a restrictive one, since the asynchronous π-calculus is included in the synchronous one.

Definition 3.1 ($\pi_{\langle\rangle}$: THE ASYNCHRONOUS π-CALCULUS WITH PAIRING) *i*) *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.

ii) *Processes* are defined by the grammar:

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

iii) We consider n bound in $(\nu n)\,P$, x bound in $a(x).P$, and x and y to be bound in the *let*-construct $\mathit{let}\,\langle x,y\rangle = p\ \mathit{in}\ P$. We call n free in P if it occurs in P and is not bound; we write $\mathit{fn}(P)$ for the set of free names in P, and write $\mathit{fn}(P,Q)$ for $\mathit{fn}(P)\cup\mathit{fn}(Q)$.

iv) We abbreviate $a(x).\mathit{let}\,\langle y,z\rangle = x\ \mathit{in}\ P$ with $x \notin \mathit{fn}(P)$ 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 $\overline{a}\ \langle c,d\rangle$.

v) We write $a \rightarrow \overline{b}$ for the *forwarder* [37] $a(w).\overline{b}\,w$ (called a *wire* in [44]).

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

Some remarks on the structure of processes should be made. Notice that data occurs only in two cases: $\overline{a}\,p$ and $\mathit{let}\,\langle x,y\rangle = p\ \mathit{in}\ P$, and that then p is either a single name, or a pair of names. This implies that we do not allow $\langle a,b\rangle.P$, nor $a(\langle b,c\rangle).P$, nor $\overline{a}\,\langle\langle b,c\rangle,d\rangle$, nor $(\nu\langle a,b\rangle)\,P$, nor $\mathit{let}\,\langle\langle a,b\rangle,y\rangle = p\ \mathit{in}\ P$, etc. Therefore substitution $P[p/x]$ is a partial operation, which depends on the places in P where x occurs.

Definition 3.2 (CONGRUENCE) The *structural congruence* is the smallest equivalence relation closed under contexts containing the following rules:

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

Because of the last clause, we will not treat $\mathit{let}\,\langle x,y\rangle = \langle a,b\rangle\ \mathit{in}\ P$ as syntactic representation of a process; this implies, for example, that we do not deal explicitly with the process $\mathit{let}\,\langle x,y\rangle = \langle a,b\rangle\ \mathit{in}\ P$ in our type assignment system.

As usual, we will consider processes modulo congruence.[12] Because of rule $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$, we will normally not write brackets in a parallel composition of more than two processes.

Definition 3.3 (Reduction) *i*) The *reduction relation* $\rightarrow_\pi$ over the processes of the $\pi_{\langle\rangle}$-calculus is defined by following (elementary) rules:

$$\begin{array}{rcll}
\overline{a}\,b \mid a(x).Q & \rightarrow & Q[b/x] & \textit{synchronisation} \\
\overline{a}\langle b,c\rangle \mid a(x).Q & \rightarrow & Q[\langle a,b\rangle/x],\ \textit{if well defined} & \\
P \rightarrow Q & \Rightarrow & (\nu n)\,P \rightarrow (\nu n)\,Q & \textit{binding} \\
P \rightarrow Q & \Rightarrow & P \mid R \rightarrow Q \mid R & \textit{composition}
\end{array}$$

ii) We write $\rightarrow_\pi^+$ for the transitive closure of $\rightarrow_\pi$, $\rightarrow_\pi^*$ for its reflexive and transitive closure; we write $\rightarrow_\pi (a)$ if we want to point out that a synchronisation took place over channel a, and write $(=_\alpha)$ if we want to point out that α-conversion has taken place.

As remarked above, $Q[p/x]$ as used in the synchronisation rule needs to be well defined for the synchronisation to take place; this implies that, for example, a synchronisation like $\overline{a}\,p \mid a(z).\overline{z}\,c$ is stuck, as is $\overline{a}\,p \mid a(z).\overline{b}\langle z,v\rangle$, etc. Note that

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

exactly as intended.

Moreover, we explicitly allow the forwarder to accept *data*, so do not enforce a fixed arity on channels.

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

Definition 3.4 (Weak bisimilarity) *i*) We write $P{\downarrow}\overline{n}$ (and say that P *outputs on* n) if $P \equiv (\nu b_1 \ldots b_m)\,(\overline{n}\,p \mid Q)$ for some Q, where $n \notin \{b_1 \ldots b_m\}$. We write $P{\Downarrow}\overline{n}$ (P *will output on* n) if there exists Q such that $P \rightarrow_\pi^* Q$ and $Q{\downarrow}n$. $P{\downarrow}n$ (P *inputs on* n) and $P{\Downarrow}n$ (P *will input on* n) are defined similarly.

ii) A *weak barbed similarity* $\sqsubseteq$ is the largest relation such that $P \sqsubseteq Q$ satisfies the following clauses:

a) if for each name n: if $P{\downarrow}\overline{n}$ then $Q{\Downarrow}\overline{n}$, and if $P{\downarrow}n$ then $Q{\Downarrow}n$;

b) for all P', if $P \rightarrow_\pi^* P'$, then there exists Q' such that $Q \rightarrow_\pi^* Q'$ and $P' \sqsubseteq Q'$.

iii) *Weak similarity* $\precsim$ is defined by: $P \precsim Q$ when $C[P] \sqsubseteq C[Q]$ for all contexts $C[\cdot]$.

iv) *Weak bisimilarity* $\approx$ is defined by: $P \approx Q$ if and only if $P \precsim Q$ and $Q \precsim P$.

Weak bisimilarity as we define here is also known as 'barbed congruence'.

The following property is standard:

Proposition 3.5 *i*) $\equiv \subseteq \precsim$.

ii) $\precsim$ *is a preorder.*

[12] For example, we need not define the rule $P \equiv Q\ \&\ Q \rightarrow Q'\ \&\ Q' \equiv P' \Rightarrow P \rightarrow P'$.

The following is easy to show, and will be used to show that synchronisations inside the image of our encodings takes place over hidden channels.

Proposition 3.6 Let P, Q not contain a and $a \neq b$, then

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

We will need the following property:

Lemma 3.7

i) Let a be at most only used as output channel in P and as input channel in Q, and the only channel that is used for input in one, and for output in the other. Then:

$$(\nu a)\,(!\,P \mid !\,Q) \;\underset{\approx}{\sqsupset}\; !\,(\nu a)\,(!\,P \mid !\,Q)$$

ii) Let x, α both be different from β, and β not used for input in Q, then:

$$(\nu x\alpha)\,(!\,Q \mid !\,\overline{\beta}\langle x,\alpha\rangle) \;\underset{\approx}{\sqsupset}\; !\,(\nu x\alpha)\,(!\,Q \mid !\,\overline{\beta}\langle x,\alpha\rangle)$$

iii) Let a be at most only used as output channel in Q and as x as input channel in R, y not used for output in either Q or R, and no other channel is used for input in one, and for output in the other. Then:

$$(\nu xa)\,(!\,Q \mid !\,y(v,d).(!\,a \rightarrow \overline{v} \mid !\,d \rightarrow \overline{x}) \mid !\,R) \;\underset{\approx}{\sqsupset}\; !\,(\nu xa)\,(!\,Q \mid !\,y(v,d).(!\,a \rightarrow \overline{v} \mid !\,d \rightarrow \overline{x}) \mid !\,R)$$

iv) Let a be at most only used as output channel in Q and as x as input channel in R, and no other channel is used for input in one, and for output in the other. Then:

$$(\nu x\alpha)\,(!\,Q \mid !\,\alpha \rightarrow \overline{x} \mid !\,R) \;\underset{\approx}{\sqsupset}\; !\,(\nu x\alpha)\,(!\,Q \mid !\,\alpha \rightarrow \overline{x} \mid !\,R)$$

Proof: For part (*i*), notice that for any context that interacts just with P or Q and does not enable synchronisation over a, the observable behaviour is the same. If a context enables synchronisation over a in $(\nu a)\,(!\,P \mid !\,Q)$ by interacting with both P and Q, then it might be that it interacts with P in $(\nu a_1)\,(!\,P \mid !\,Q)$ and with Q in $(\nu a_2)\,(!\,P \mid !\,Q)$, thus not enabling the synchronisation. So $!\,(\nu a)\,(!\,P \mid !\,Q)$ has less observable behaviour.

The other cases are similar. □

4 A natural translation for $\mathcal{LK}$ into $\pi_{\langle\rangle}$ that respects head reduction

In this section we will present a first translation $[\![\cdot]\!]_{\mathrm{N}}$ of $\mathcal{LK}$ into $\pi_{\langle\rangle}$; it is called natural since it will create processes that output on names that are associated to plugs, and input on names that are associated to sockets, and tries as much as possible to encode the joining of connectors through substitution, following the syntactic structure of terms. Also, it is natural since we can show that head reduction $\mathcal{LK}$ is implemented through synchronisation in $\pi_{\langle\rangle}$ (see Theorem 4.8); for the semantic translation of the next section we can show that reduction in $\mathcal{LK}$ is represented through weak bisimilarity (see Theorem 5.4).

Although departing from $\mathcal{LK}$ 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.

We will first give the definition of the natural translation, and then explain the details.

Definition 4.1 (Natural translation of $\mathcal{LK}$ in $\pi_{\langle\rangle}$) The *natural* translation is defined by:

$$\begin{array}{rcl}
\llbracket \langle x{\cdot}\beta\rangle \rrbracket_{\mathrm{N}} &=& x(w).\overline{\beta}w \\
\llbracket \widehat{z}P\widehat{\alpha}{\cdot}\beta \rrbracket_{\mathrm{N}} &=& (\nu z\alpha)\,(!\,\llbracket P \rrbracket_{\mathrm{N}} \mid \overline{\beta}\langle z,\alpha\rangle) \\
\llbracket P\widehat{\alpha}\,[x]\,\widehat{z}Q \rrbracket_{\mathrm{N}} &=& x(\alpha,z).(!\,\llbracket P \rrbracket_{\mathrm{N}} \mid !\,\llbracket Q \rrbracket_{\mathrm{N}}) \\
\llbracket P\widehat{\alpha} \dagger \widehat{z}Q \rrbracket_{\mathrm{N}} &=& (\nu z)\,(!\,\llbracket P \rrbracket_{\mathrm{N}}[z/\alpha] \mid !\,\llbracket Q \rrbracket_{\mathrm{N}})
\end{array}$$

Let us investigate the intuition behind this definition for a moment. The interpretation of $P\widehat{\alpha} \dagger \widehat{z}Q$ will generate a process $\llbracket P \rrbracket_{\mathrm{N}}$ that (possibly) outputs on α, and $\llbracket Q \rrbracket$ that inputs on z; since the intention of the *cut* is that α and z are connected, we realise this directly by renaming α by z:[13]

$$\llbracket P\widehat{\alpha} \dagger \widehat{z}Q \rrbracket_{\mathrm{N}} \;=\; (\nu z)\,(!\,\llbracket P \rrbracket_{\mathrm{N}}[z/\alpha] \mid !\,\llbracket Q \rrbracket_{\mathrm{N}})$$

Since z (and α) are bound in $P\widehat{\alpha} \dagger \widehat{z}Q$, z is restricted.

Likewise, when constructing the process that represents the term $P\widehat{\alpha}\,[x]\,\widehat{z}Q$, we will generate a process $\llbracket P \rrbracket_{\mathrm{N}}$ that outputs on α, and $\llbracket Q \rrbracket_{\mathrm{N}}$ that inputs on z. Notice that when that term is placed in a cut with an export

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

a reduction step can be made that generates the term $P\widehat{\alpha} \dagger \widehat{y}(R\widehat{\beta} \dagger \widehat{z}Q)$. Therefore, we can see the synchronisation over x as enabling the connection of α to y and β to z. So, in effect, then the synchronisation over x exchanges *two* names (hence the need for pairing), and we can see x in $\llbracket P\widehat{\alpha}\,[x]\,\widehat{z}Q \rrbracket_{\mathrm{N}}$ as an input that receives the names (here y and β) that will be connected to the names α and z, which in [7] we defined as:

$$x(v,d)((\nu\alpha)\,(!\,\llbracket P \rrbracket_{\mathrm{N}} \mid !\,\alpha{\rightarrow}\overline{v}) \mid (\nu z)\,(!\,d{\rightarrow}\overline{z} \mid !\,\llbracket Q \rrbracket_{\mathrm{N}}))$$

Here we can improve on that naturally by treating α and z as *variables*. We place the interpretations of P and Q in parallel and introduce a guard using x, that receives on x the channel names that are going to be substituted for α and z:

$$\llbracket P\widehat{\alpha}\,[x]\,\widehat{z}Q \rrbracket_{\mathrm{N}} \;=\; x(\alpha,z).(!\,\llbracket P \rrbracket_{\mathrm{N}} \mid !\,\llbracket Q \rrbracket_{\mathrm{N}})$$

This then causes the use of pairing in the interpretation of the export $\widehat{z}P\widehat{\alpha}{\cdot}\beta$ as well. When we see $\llbracket R \rrbracket_{\mathrm{N}}$ as a process that can input on z and output on α, then $\llbracket \widehat{z}P\widehat{\alpha}{\cdot}\beta \rrbracket_{\mathrm{N}}$ is the process that communicates that fact over β, by sending the two names:

$$\llbracket \widehat{z}P\widehat{\alpha}{\cdot}\beta \rrbracket_{\mathrm{N}} \;=\; (\nu z\alpha)\,(!\,\llbracket P \rrbracket_{\mathrm{N}} \mid \overline{\beta}\langle z,\alpha\rangle)$$

since z and α are bound in $\widehat{z}P\widehat{\alpha}{\cdot}\beta$, they are restricted.

Remark 4.2 We can now better illustrate why we have not used an output guard on the interpretation of exports $\widehat{y}P\widehat{\beta}{\cdot}\alpha$ as in $(\nu y\beta)\,(\overline{\alpha}\langle y,\beta\rangle.!\llbracket P \rrbracket)$, as might have been expected, but have placed the communication of the interface in parallel as in $(\nu y\beta)\,(!\,\llbracket P \rrbracket_{\mathrm{N}} \mid \overline{\alpha}\langle y,\beta\rangle)$. Take the term $(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle$, which by rule (exp) reduces to $\widehat{y}P\widehat{\beta}{\cdot}\gamma$. We would want the interpretation of the first term to at least include (in terms of $\mathrel{\underset{\sim}{\sqsupseteq}}$) that of the second. Assume we would have defined

$$\llbracket \widehat{y}P\widehat{\beta}{\cdot}\alpha \rrbracket \;=\; (\nu y\beta)\,(\overline{\alpha}\langle y,\beta\rangle.!\llbracket P \rrbracket)$$

then we can only show:

[13] Notice that $(\nu z)\,(!\,\llbracket P \rrbracket_{\mathrm{N}}[z/\alpha] \mid !\,\llbracket Q \rrbracket_{\mathrm{N}}) =_{\alpha} (\nu\alpha)\,(!\,\llbracket P \rrbracket_{\mathrm{N}} \mid !\,\llbracket Q \rrbracket_{\mathrm{N}}[\alpha/z])$ since z does not occur free in P and α not in Q; we will not distinguish these and swap between them whenever convenient.

$$\begin{array}{ll}
[\![(\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x\cdot\gamma\rangle]\!] & \triangleq \\
(\nu x)\,(\,!\,(\nu y\beta)\,(\overline{x}\langle y,\beta\rangle.![\![P]\!])\mid !\,x(w).\overline{\gamma}\,w) & \equiv, \triangleq \\
(\nu x)\,((\nu y\beta)\,(\overline{x}\langle y,\beta\rangle.![\![P]\!])\mid x(w).\overline{\gamma}\,w \mid ![\![\widehat{y}P\widehat{\beta}\cdot x]\!]\mid !\,x(w).\overline{\gamma}\,w) & \to_\pi (x) \\
(\nu y\beta)\,(\,![\![P]\!]\mid\overline{\gamma}\langle y,\beta\rangle)\mid[\![(\widehat{y}P\widehat{\beta}\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}\langle x\cdot\gamma\rangle]\!] &
\end{array}$$

Notice that this last process then does not include $[\![\widehat{y}P\widehat{\beta}\cdot\gamma]\!]$; in fact, $(\nu y\beta)\,(\overline{\gamma}\langle y,\beta\rangle.![\![P]\!])$ and $(\nu y\beta)\,(\,![\![P]\!]\mid\overline{\gamma}\langle y,\beta\rangle)$ are not even weakly bisimilar. So we are forced to place the output $\overline{\alpha}\langle y,\beta\rangle$ in parallel to $[\![P]\!]_{\mathrm{N}}$ in $[\![\widehat{y}P\widehat{\beta}\cdot\alpha]\!]_{\mathrm{N}}$.

Remark 4.3 (On confluence and $\sqsupseteq$) As observed in Remark 2.7, the *cut* $P\widehat{\alpha}\dagger\widehat{x}Q$ – with α not in P and x not in Q – in $\mathcal{LK}$ runs via erasure to either P or Q, and reducing it decreases the set of reachable normal forms. Observe that in the image of $\mathcal{LK}$ in π, being built without using 'choice', there is no notion of *erasure* of processes; this implies that, using reduction in the π-calculus, we cannot model $P\widehat{\alpha}\dagger\widehat{x}Q \to_{\mathcal{LK}} P$; we can at most show:

$$[\![P\widehat{\alpha}\dagger\widehat{x}Q]\!]_{\mathrm{N}} \;\triangleq\; (\nu x)\,(\,![\![P]\!]_{\mathrm{N}}[x/\alpha]\mid ![\![Q]\!]_{\mathrm{N}}) \;\equiv\; ![\![P]\!]_{\mathrm{N}}\mid ![\![Q]\!]_{\mathrm{N}}$$

assuming $\alpha\notin fp(P)$ and $x\notin fs(Q)$. Now all reductions will take place in either $[\![P]\!]_{\mathrm{N}}$ or $[\![Q]\!]_{\mathrm{N}}$, and both parts will remain under reduction. This implies that, in this case, it is clear that the interpreted *cut* $[\![P\widehat{\alpha}\dagger\widehat{x}Q]\!]_{\mathrm{N}}$ must *contain* the behaviour of either its contractea, so, evidently, has more behaviour than both $[\![P]\!]_{\mathrm{N}}$ and $[\![Q]\!]_{\mathrm{N}}$ separately. As stated in Remark 2.7, this is natural for translations of non-confluent calculi, since there $P\to_{\mathcal{LK}} Q$ implies $[\![Q]\!]\subseteq[\![P]\!]$. We see this return in the formulation of the correctness result (Theorem 4.8) for the natural translation, which is formulated through the relation $\sqsupseteq$.

Since in this translation some sub-terms are placed under input, a full representation of reduction in $\mathcal{LK}$ cannot be achieved: it is not possible to reduce the (interpreted) terms that appear under an input. To accommodate for this shortcoming, to achieve a simulation result using this first translation, we have to restrict the notion of reduction on $\mathcal{LK}$ to that of *head* reduction. In view of the literature that exists on translations into the π-calculus, this is unfortunate but standard: the encoding forces a restriction on the modelled reduction rules. A similar limitation was already evident for Milner's encoding of the λ-calculus in [42], which manages only to show a preservation result for (large step) *lazy* reduction [3] for the λ-calculus, and is thereby also present in all research that is based on that approach; also [11] has to limit the notion of reduction to spine reduction, and [12] to head reduction.

As can be seen in Definition 4.1, input is used for the translation of *import*, so the restriction will consist of removing the rules that reduce under *import*; notice that this forces the exclusion of the second and third contextual rule, as well as the propagation rules $(imp\dagger)$, $(\dagger imp\text{-}out)$, and $(\dagger imp\text{-}in)$.

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

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

The restriction we put on the rewriting system in head-reduction implies that we can only contract a *cut* $T\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R)$ if T is a term with α introduced; as observed above, we can compare this term, with discrepancies, to $TQ\overrightarrow{R_i}$ (where R is the context $[\,]\overrightarrow{R_i}$). In particular, under head-reduction, in the term $T\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R)$ all reduction takes place exclusively *inside* T (so in the head of the term $TQ\overrightarrow{R_i}$), and the *cut* mentioned explicitly will only be contracted after that reduction produces a term that introduces α in an *export*. Moreover, even when $T\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R)$ reduces to $(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R)$, we can continue running inside P. In any case, the contraction of this *cut* 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 head introduces α. Notice that head reduction in $\mathcal{LK}$ models more than just what we suggest here: continuing on the metaphor of the λ-calculus, the *cut* $P\widehat{\alpha} \dagger \widehat{x}Q$ corresponds to $Q\langle x := P\rangle$; head reduction allows reduction in *both* components of the *cut*, so allows for reduction *during* substitution.

Following on from these observations, it is clear that in terms of the representation of computable functions, head reduction is still fully expressive. In fact, the spine translation of [11] is a combination of the mapping of the λ-calculus into $\mathcal{X}$ (or of natural deduction into the sequent calculus) and the natural translation we define here; a similar observation can be made with respect to the interpretation of $\Lambda\mu$ and $\lambda\mu$ presented in [12, 13]. In that paper it is shown that explicit λ-spine reduction is preserved step-by-step by the induced combination of $\mathcal{LK}$'s head reduction and π's synchronisation; it also shows that typeability is preserved.

Remark 4.4 We can make the following observations:

- The synchronisations generated by the natural translation only involve processes of the shape:
$$x(w).\overline{\alpha}\,w \qquad \overline{\beta}\langle x,\alpha\rangle \qquad z(\beta,y).(P \mid Q)$$
so in particular, substitution $P[p/x]$ as generated by synchronisation is always well defined. These synchronisations are of the shape:
  - $(\nu x)\,((\nu y\beta)\,(P \mid \overline{x}\langle y,\beta\rangle) \mid x(\alpha,z).(R \mid Q)) \ \rightarrow_\pi\ (\nu y\beta)\,(P \mid R[y/\alpha] \mid Q[\beta/z])$, and after the synchronisation over x, P can receive over y from $R[y/\alpha]$ and send over β to $Q[\beta/z]$; or
  - $(\nu x)\,((\nu y\beta)\,(P \mid \overline{x}\langle y,\beta\rangle) \mid x(w).\overline{\alpha}\,w) \ \rightarrow_\pi\ (\nu y\beta)\,(P \mid \overline{\alpha}\langle y,\beta\rangle)$.
- All synchronisation takes place *only* over channels whose names are bound connectors in the terms that are interpreted. In particular,
  - no synchronisation is possible in $[\![P\widehat{\alpha} \dagger \widehat{x}Q]\!]_{\mathrm{N}} \triangleq (\nu x)\,(!\,[\![P]\!]_{\mathrm{N}}[x/\alpha] \mid !\,[\![Q]\!]_{\mathrm{N}})$ between and $[\![P]\!]_{\mathrm{N}}[x/\alpha]$ and $[\![Q]\!]_{\mathrm{N}}$ but over channel x; and
  - no direct synchronisation is possible in $[\![P\widehat{\alpha}\,[x]\,\widehat{y}Q]\!]_{\mathrm{N}} \triangleq x(\alpha,y).(!\,[\![P]\!]_{\mathrm{N}} \mid !\,[\![Q]\!]_{\mathrm{N}})$ between $[\![P]\!]_{\mathrm{N}}$ and $[\![Q]\!]_{\mathrm{N}}$, even after input has been received over x.
- The translation is not trivial, since
$$\begin{aligned}[\![\widehat{y}(\widehat{z}\langle y{\cdot}\beta\rangle\,\widehat{\beta}{\cdot}\gamma)\,\widehat{\gamma}{\cdot}\alpha]\!]_{\mathrm{N}} &= (\nu y\gamma)\,(!\,(\nu z\beta)\,(!\,y(w).\overline{\beta}\,w \mid \overline{\gamma}\langle z,\beta\rangle) \mid \overline{\alpha}\langle y,\gamma\rangle)\\ [\![\widehat{x}\langle x{\cdot}\delta\rangle\,\widehat{\delta}{\cdot}\alpha]\!]_{\mathrm{N}} &= (\nu x\delta)\,(!\,x(w).\overline{\delta}\,w \mid \overline{\alpha}\langle x,\delta\rangle)\end{aligned}$$
(witnesses of, respectively, $\vdash A{\to}B{\to}A$ and $\vdash C{\to}C$) yielding processes that differ under $\approx$.

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 translation to respect reduction, in particular we need to be able to reduce the translation 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 interpret the *export* of x and α over β as $\overline{\beta}\langle x,\alpha\rangle$ would force the translation of $\langle z{\cdot}\gamma\rangle$ to always receive a pair of names. But requiring for the translation of a *capsule* to always deal with pairs of names is too restrictive: we will see that then only arrow types could be assigned. Therefore it is desirable to allow those 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.

Example 4.5 The translation of $\llbracket \widehat{z}((\widehat{y}\langle y{\cdot}\delta\rangle\, \widehat{\eta}{\cdot}\alpha)\widehat{\alpha}\,[z]\,\widehat{v}\langle v{\cdot}\delta\rangle)\widehat{\delta}{\cdot}\gamma \rrbracket_{\mathrm{N}}$, the witness of Peirce's law of Example 1.5, becomes:

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

That this process is a witness of $\vdash((A{\to}B){\to}A){\to}A$ is a straightforward application of Theorem 6.8.

The following is straightforward:

Proposition 4.6 (Free name preservation) *$\alpha,x \notin fc(P)$, if and only if $\alpha,x \notin fn(\llbracket P \rrbracket_{\mathrm{N}})$.*

We will show in Theorem 4.8 that we can mimic $\mathcal{LK}$'s head reduction in $\pi_{\langle\rangle}$: if $P \to_{\mathrm{H}} Q$, the image of the $\mathcal{LK}$-term P under the translation in $\pi_{\langle\rangle}$ reduces to some $\pi_{\langle\rangle}$-process that contains the behaviour of Q, but might have some extra behaviour as well. As will become clear also in the proofs below, this is in part due to the presence of replicated processes in the translation of the *cut*, but also comes from the fact that reduction in $\mathcal{LK}$ is not confluent, as discussed in Remark 4.3.

First we need to show the following:

Lemma 4.7 i) Assume γ does not occur free in $P[\alpha/x]$, and x can only be a free socket in P, then:

$$\begin{array}{ll} (\nu\alpha)\,(!\,(\nu y\beta)\,(!\,\llbracket Q \rrbracket_{\mathrm{N}} \mid \overline{\gamma}\langle y,\beta\rangle) \mid !\,\llbracket P \rrbracket_{\mathrm{N}}[\alpha/x]) & \underset{\approx}{\sqsupset} \\ \quad (\nu y\beta)\,(!\,(\nu\alpha)\,(!\,\llbracket Q \rrbracket_{\mathrm{N}} \mid !\,\llbracket P \rrbracket_{\mathrm{N}}[\alpha/x]) \mid \overline{\gamma}\langle y,\beta\rangle) & \end{array}$$

ii) Assume α can only be a plug in Q, and x can only be a socket in P, then:

$$\begin{array}{ll} (\nu x)\,(!\,(\nu y\beta)\,(!\,\llbracket Q \rrbracket_{\mathrm{N}}[x/\alpha] \mid \overline{x}\langle y,\beta\rangle) \mid !\,\llbracket P \rrbracket_{\mathrm{N}}) & \approx \\ \quad (\nu\gamma)\,(!\,(\nu y\beta)\,(!\,(\nu x)\,(!\,\llbracket Q \rrbracket_{\mathrm{N}}[x/\alpha] \mid !\,\llbracket P \rrbracket_{\mathrm{N}}) \mid \overline{\gamma}\langle y,\beta\rangle) \mid !\,\llbracket P \rrbracket_{\mathrm{N}}[\gamma/x]) & \end{array}$$

iii) Assume x can only be a socket in P, and α is only a plug in Q or R, then:

$$\begin{array}{ll} (\nu\alpha)\,(!\,(\nu y)\,(!\,\llbracket Q \rrbracket_{\mathrm{N}}[y/\beta] \mid !\,\llbracket R \rrbracket_{\mathrm{N}}) \mid !\,\llbracket P \rrbracket_{\mathrm{N}}[\alpha/x]) & \approx \\ \quad (\nu y)\,(!\,(\nu\alpha)\,(!\,\llbracket Q \rrbracket_{\mathrm{N}}[y/\beta] \mid !\,\llbracket P \rrbracket_{\mathrm{N}}[\alpha/x]) \mid !\,(\nu\alpha)\,(!\,\llbracket R \rrbracket_{\mathrm{N}} \mid !\,\llbracket P \rrbracket_{\mathrm{N}}[\alpha/x])) & \end{array}$$

Proof: *i*) Notice that γ and y do not occur in $\llbracket P \rrbracket_{\mathrm{N}}[\alpha/x]$. Therefore

$$\begin{array}{lll} (\nu\alpha)\,(!\,(\nu y\beta)\,(!\,\llbracket Q \rrbracket_{\mathrm{N}} \mid \overline{\gamma}\langle y,\beta\rangle) \mid !\,\llbracket P \rrbracket_{\mathrm{N}}[\alpha/x]) & \underset{\approx}{\sqsupset} & \\ (\nu\alpha)\,((\nu y\beta)\,(!\,\llbracket Q \rrbracket_{\mathrm{N}} \mid \overline{\gamma}\langle y,\beta\rangle) \mid !\,\llbracket P \rrbracket_{\mathrm{N}}[\alpha/x]) & \equiv & (\alpha \neq \gamma, y,\beta \notin \llbracket P \rrbracket_{\mathrm{N}}[\alpha/x]) \\ (\nu y\beta)\,((\nu\alpha)\,(!\,\llbracket Q \rrbracket_{\mathrm{N}} \mid !\,\llbracket P \rrbracket_{\mathrm{N}}[\alpha/x])) \mid \overline{\gamma}\langle y,\beta\rangle & \underset{\approx}{\sqsupset} & (3.7(i)) \\ (\nu y\beta)\,(!\,(\nu\alpha)\,(!\,\llbracket Q \rrbracket_{\mathrm{N}} \mid !\,\llbracket P \rrbracket_{\mathrm{N}}[\alpha/x]) \mid \overline{\gamma}\langle y,\beta\rangle) & & \end{array}$$

ii) Notice that x does not occur in Q, might occur as *socket* in P, and that α is a *plug* in Q; therefore x is only used for output in $\llbracket Q \rrbracket_{\mathrm{N}}[x/\alpha]$. We observe that

$$(\nu x)\,(!\,(\nu y\beta)\,(!\,\llbracket Q \rrbracket_{\mathrm{N}}[x/\alpha] \mid \overline{x}\langle y,\beta\rangle) \mid !\,\llbracket P \rrbracket_{\mathrm{N}} \mid !\,\llbracket P \rrbracket_{\mathrm{N}})$$

and

$$(\nu\gamma)\,((\nu x)\,(!\,(\nu y\beta)\,(!\,\llbracket Q \rrbracket_{\mathrm{N}}[x/\alpha] \mid \overline{\gamma}\langle y,\beta\rangle) \mid !\,\llbracket P \rrbracket_{\mathrm{N}}) \mid !\,\llbracket P \rrbracket_{\mathrm{N}}[\gamma/x])$$

are weakly bisimilar since the substitution of γ for x does not introduce any transition in the term ($\overline{x}\langle y,\beta\rangle$ and $\overline{\gamma}\langle y,\beta\rangle$ could communicate only with $!\,\llbracket P \rrbracket_{\mathrm{N}}$ and $!\,\llbracket P \rrbracket_{\mathrm{N}}[\gamma/x]$ respectively) and is not restricting other transitions ($!\,\llbracket Q \rrbracket_{\mathrm{N}}[x/\alpha]$ can only communicate with $!\,\llbracket P \rrbracket_{\mathrm{N}}$). Therefore:

$$\begin{array}{ll}
(\nu x)\,(!\,(\nu y\beta)\,(!\,\llbracket Q\rrbracket_{\mathrm{N}}[x/\alpha]\mid \overline{x}\langle y,\beta\rangle)\mid !\,\llbracket P\rrbracket_{\mathrm{N}}) & \approx \\
(\nu x)\,(!\,(\nu y\beta)\,(!\,\llbracket Q\rrbracket_{\mathrm{N}}[x/\alpha]\mid \overline{x}\langle y,\beta\rangle)\mid !\,\llbracket P\rrbracket_{\mathrm{N}}\mid !\,\llbracket P\rrbracket_{\mathrm{N}}) & \approx\ (=_{\alpha}, \gamma\textit{ fresh}) \\
(\nu \gamma)\,((\nu x)\,(!\,(\nu y\beta)\,(!\,\llbracket Q\rrbracket_{\mathrm{N}}[x/\alpha]\mid \overline{\gamma}\langle y,\beta\rangle)\mid !\,\llbracket P\rrbracket_{\mathrm{N}})\mid !\,\llbracket P\rrbracket_{\mathrm{N}}[\gamma/x]) & \approx\ (\textit{as in (i)}) \\
(\nu \gamma)\,(!\,(\nu y\beta)\,(!\,(\nu x)\,(!\,\llbracket Q\rrbracket_{\mathrm{N}}[x/\alpha]\mid !\,\llbracket P\rrbracket_{\mathrm{N}})\mid \overline{\gamma}\langle y,\beta\rangle)\mid !\,\llbracket P\rrbracket_{\mathrm{N}}[\gamma/x]) &
\end{array}$$

iii) Notice that α is not used for input in either $\llbracket R\rrbracket_{\mathrm{N}}$ or $\llbracket Q\rrbracket_{\mathrm{N}}[y/\beta]$. The processes

$$(\nu\alpha)\,(!\,(\nu y)\,(!\,\llbracket Q\rrbracket_{\mathrm{N}}[y/\beta]\mid !\,\llbracket R\rrbracket_{\mathrm{N}})\mid !\,\llbracket P\rrbracket_{\mathrm{N}}[\alpha/x]\mid !\,\llbracket P\rrbracket_{\mathrm{N}}[\alpha/x])$$

and

$$(\nu y)\,(!\,(\nu\alpha)\,(!\,\llbracket Q\rrbracket_{\mathrm{N}}[y/\beta]\mid !\,\llbracket P\rrbracket_{\mathrm{N}}[\alpha/x])\mid !\,(\nu\alpha)\,(!\,\llbracket R\rrbracket_{\mathrm{N}}\mid !\,\llbracket P\rrbracket_{\mathrm{N}}[\alpha/x]))$$

are weakly bisimilar for a reasoning similar to the one above. Therefore:

$$\begin{array}{ll}
(\nu\alpha)\,(!\,(\nu y)\,(!\,\llbracket Q\rrbracket_{\mathrm{N}}[y/\beta]\mid !\,\llbracket R\rrbracket_{\mathrm{N}})\mid !\,\llbracket P\rrbracket_{\mathrm{N}}[\alpha/x]) & \approx \\
(\nu\alpha)\,(!\,(\nu y)\,(!\,\llbracket Q\rrbracket_{\mathrm{N}}[y/\beta]\mid !\,\llbracket R\rrbracket_{\mathrm{N}})\mid !\,\llbracket P\rrbracket_{\mathrm{N}}[\alpha/x]\mid !\,\llbracket P\rrbracket_{\mathrm{N}}[\alpha/x]) & \approx \\
(\nu y)\,(!\,(\nu\alpha)\,(!\,\llbracket Q\rrbracket_{\mathrm{N}}[y/\beta]\mid !\,\llbracket P\rrbracket_{\mathrm{N}}[\alpha/x])\mid !\,(\nu\alpha)\,(!\,\llbracket R\rrbracket_{\mathrm{N}}\mid !\,\llbracket P\rrbracket_{\mathrm{N}}[\alpha/x])) &
\end{array}$$

We can show the following correctness result for head reduction, $\rightarrow_{\mathrm{H}}$:

Theorem 4.8 (Operational Soundness of $\llbracket\cdot\rrbracket_{\mathrm{N}}$ with respect to $\rightarrow_{\mathrm{H}}$) *If $P \rightarrow^{*}_{\mathrm{H}} Q$, then there exists R such that $\llbracket P\rrbracket_{\mathrm{N}} \rightarrow^{*}_{\pi} R$ with $R \gtrsim \llbracket Q\rrbracket_{\mathrm{N}}$.*

Proof: By induction on the definition of reduction. We only show the more illustrative cases, and deal with the rules in the order they were presented in Section 2.

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

$$\begin{array}{lll}
\llbracket\langle y{\cdot}\alpha\rangle\widehat{\alpha}\dagger\widehat{x}\langle x{\cdot}\gamma\rangle\rrbracket_{\mathrm{N}} & \triangleq & (\nu x)\,(!\,y(w).\overline{x}\,w\mid !\,x(w).\overline{\gamma}\,w)\ \approx\ !\,y(w).\overline{\gamma}\,w\ \triangleq \\
!\,\llbracket\langle y{\cdot}\gamma\rangle\rrbracket_{\mathrm{N}} & \gtrsim & \llbracket\langle y{\cdot}\gamma\rangle\rrbracket_{\mathrm{N}}
\end{array}$$

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

$$\begin{array}{ll}
\llbracket(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha}\dagger\widehat{x}\langle x{\cdot}\gamma\rangle\rrbracket_{\mathrm{N}} & \triangleq \\
(\nu x)\,(!\,(\nu y\beta)\,(!\,\llbracket P\rrbracket_{\mathrm{N}}\mid \overline{x}\langle y,\beta\rangle)\mid !\,x(w).\overline{\gamma}\,w) & \equiv \\
(\nu x)\,((\nu y\beta)\,(!\,\llbracket P\rrbracket_{\mathrm{N}}\mid \overline{x}\langle y,\beta\rangle)\mid x(w).\overline{\gamma}\,w\mid !\,(\nu y\beta)\,(!\,\llbracket P\rrbracket_{\mathrm{N}}\mid \overline{x}\langle y,\beta\rangle)\mid !\,x(w).\overline{\gamma}\,w) & \rightarrow_{\pi}(x) \\
(\nu y\beta)\,(!\,\llbracket P\rrbracket_{\mathrm{N}}\mid \overline{\gamma}\langle y,\beta\rangle)\mid \llbracket(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha}\dagger\widehat{x}\langle x{\cdot}\gamma\rangle\rrbracket_{\mathrm{N}} & \triangleq \\
\llbracket\widehat{y}P\widehat{\beta}{\cdot}\gamma\rrbracket_{\mathrm{N}}\mid \llbracket(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha}\dagger\widehat{x}\langle x{\cdot}\gamma\rangle\rrbracket_{\mathrm{N}} & \gtrsim\ \llbracket\widehat{y}P\widehat{\beta}{\cdot}\gamma\rrbracket_{\mathrm{N}}
\end{array}$$

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

$$\begin{array}{lll}
\llbracket\langle y{\cdot}\alpha\rangle\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{z}P)\rrbracket_{\mathrm{N}} & \triangleq & (\nu x)\,(!\,y(w).\overline{x}\,w\mid !\,x(\beta,z).(!\,\llbracket Q\rrbracket_{\mathrm{N}}\mid !\,\llbracket R\rrbracket_{\mathrm{N}}))\ \approx \\
!\,y(\beta,z).(!\,\llbracket Q\rrbracket_{\mathrm{N}}\mid !\,\llbracket R\rrbracket_{\mathrm{N}}) & \triangleq & !\,\llbracket Q\widehat{\beta}\,[y]\,\widehat{z}R\rrbracket_{\mathrm{N}}\ \gtrsim\ \llbracket Q\widehat{\beta}\,[y]\,\widehat{z}R\rrbracket_{\mathrm{N}}
\end{array}$$

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

$$\begin{array}{ll}
\llbracket(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R)\rrbracket_{\mathrm{N}} & \triangleq \\
(\nu x)\,(!\,\llbracket\widehat{y}P\widehat{\beta}{\cdot}x\rrbracket_{\mathrm{N}}\mid !\,\llbracket Q\widehat{\gamma}\,[x]\,\widehat{z}R\rrbracket_{\mathrm{N}}) & \triangleq \\
(\nu x)\,(!\,(\nu y\beta)\,(!\,\llbracket P\rrbracket_{\mathrm{N}}\mid \overline{x}\langle y,\beta\rangle)\mid !\,x(\gamma,z).(!\,\llbracket Q\rrbracket_{\mathrm{N}}\mid !\,\llbracket R\rrbracket_{\mathrm{N}})) & \equiv,\triangleq \\
(\nu x)\,((\nu y\beta)\,(!\,\llbracket P\rrbracket_{\mathrm{N}}\mid \overline{x}\langle y,\beta\rangle)\mid x(\gamma,z).(!\,\llbracket Q\rrbracket_{\mathrm{N}}\mid !\,\llbracket R\rrbracket_{\mathrm{N}})\mid & \\
\qquad\qquad !\,\llbracket\widehat{y}P\widehat{\beta}{\cdot}x\rrbracket_{\mathrm{N}}\mid !\,\llbracket Q\widehat{\gamma}\,[x]\,\widehat{z}R\rrbracket_{\mathrm{N}}) & \rightarrow_{\pi}\ (x) \\
(\nu y\beta)\,(!\,\llbracket P\rrbracket_{\mathrm{N}}\mid !\,\llbracket Q\rrbracket_{\mathrm{N}}[y/\gamma]\mid !\,\llbracket R\rrbracket_{\mathrm{N}}[\beta/z])\ \mid\ \llbracket(\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha}\dagger\widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R)\rrbracket_{\mathrm{N}} & \equiv,\gtrsim \\
(\nu y)\,(!\,\llbracket Q[y/\gamma]\rrbracket_{\mathrm{N}}\mid (\nu\beta)\,(!\,\llbracket P\rrbracket_{\mathrm{N}}\mid !\,\llbracket R[\beta/z]\rrbracket_{\mathrm{N}})) & \gtrsim\ \ (3.7(i)) \\
(\nu y)\,(!\,\llbracket Q\rrbracket_{\mathrm{N}}[y/\gamma]\mid !\,(\nu\beta)\,(!\,\llbracket P\rrbracket_{\mathrm{N}}\mid !\,\llbracket R\rrbracket_{\mathrm{N}}[\beta/z])) & \triangleq \\
(\nu y)\,(!\,\llbracket Q\rrbracket_{\mathrm{N}}[y/\gamma]\mid !\,\llbracket P\widehat{\beta}\dagger\widehat{z}R\rrbracket_{\mathrm{N}}) & \triangleq \\
\llbracket Q\widehat{\gamma}\dagger\widehat{y}(P\widehat{\beta}\dagger\widehat{z}R)\rrbracket_{\mathrm{N}} &
\end{array}$$

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, since

$$\begin{array}{lll}
(\nu y\beta)\,(!\,[\![P]\!]_{\mathrm N} \mid !\,[\![Q]\!]_{\mathrm N}[y/\gamma] \mid !\,[\![R]\!]_{\mathrm N}[\beta/z]) & \equiv & \\
(\nu \beta)\,((\nu y)\,(!\,[\![Q[y/\gamma]]\!]_{\mathrm N} \mid !\,[\![P]\!]_{\mathrm N}) \mid !\,[\![R[\beta/z]]\!]_{\mathrm N}) & \underset{\sim}{\sqsupset} & (3.7(i)) \\
(\nu \beta)\,(!\,(\nu y)\,(!\,[\![Q]\!]_{\mathrm N}[y/\gamma] \mid !\,[\![P]\!]_{\mathrm N}) \mid !\,[\![R]\!]_{\mathrm N}[\beta/z]) & \triangleq & \\
(\nu \beta)\,(!\,[\![Q\widehat{\gamma} \dagger \widehat{y}P]\!]_{\mathrm N} \mid !\,[\![R]\!]_{\mathrm N}[\beta/z]) & \triangleq & [\![(Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R]\!]_{\mathrm N}
\end{array}$$

Left propagation: $(cap\dagger) : \langle y{\cdot}\beta\rangle\,\widehat{\alpha} \dagger \widehat{x}P \to \langle y{\cdot}\beta\rangle,\ \beta \neq \alpha$:

$$\begin{array}{l}
[\![\langle y{\cdot}\beta\rangle\,\widehat{\alpha} \dagger \widehat{x}P]\!]_{\mathrm N} \quad \triangleq \quad (\nu x)\,(!\,[\![\langle y{\cdot}\beta\rangle]\!]_{\mathrm N} \mid !\,[\![P]\!]_{\mathrm N}) \ \equiv \\
!\,[\![\langle y{\cdot}\beta\rangle]\!]_{\mathrm N} \mid (\nu x)\,(!\,[\![P]\!]_{\mathrm N}) \ \underset{\sim}{\sqsupset}\ [\![\langle y{\cdot}\beta\rangle]\!]_{\mathrm N}
\end{array}$$

$(exp\text{-}out\dagger) : (\widehat{y}Q\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}P \to (\widehat{y}(Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{x}P,\ \gamma\ fresh$:

$$\begin{array}{lll}
[\![(\widehat{y}Q\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}P]\!]_{\mathrm N} & \triangleq & \\
(\nu x)\,(!\,(\nu y\beta)\,(!\,[\![Q]\!]_{\mathrm N}[x/\alpha] \mid \overline{x}\langle y,\beta\rangle) \mid !\,[\![P]\!]_{\mathrm N}) & \approx & (4.7(ii)) \\
(\nu \gamma)\,(!\,(\nu y\beta)\,(!\,(\nu x)\,(!\,[\![Q]\!]_{\mathrm N}[x/\alpha] \mid !\,[\![P]\!]_{\mathrm N}) \mid \overline{\gamma}\langle y,\beta\rangle) \mid !\,[\![P]\!]_{\mathrm N}[\gamma/x]) & \triangleq & \\
[\![(\widehat{y}(Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{x}P]\!]_{\mathrm N} & &
\end{array}$$

$(exp\text{-}in\dagger)$: $(\widehat{y}Q\widehat{\beta}{\cdot}\gamma)\widehat{\alpha} \dagger \widehat{x}P \to \widehat{y}(Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma, \gamma \neq \alpha.$

$$\begin{array}{lll}
[\![(\widehat{y}Q\widehat{\beta}{\cdot}\gamma)\widehat{\alpha} \dagger \widehat{x}P]\!]_{\mathrm N} & \triangleq & \\
(\nu \alpha)\,(!\,(\nu y\beta)\,(!\,[\![Q]\!]_{\mathrm N} \mid \overline{\gamma}\langle y,\beta\rangle) \mid !\,[\![P]\!]_{\mathrm N}[\alpha/x]) & \underset{\sim}{\sqsupset} & (4.7(i)) \\
(\nu y\beta)\,(!\,(\nu \alpha)\,(!\,[\![Q]\!]_{\mathrm N} \mid !\,[\![P]\!]_{\mathrm N}[\alpha/x]) \mid \overline{\gamma}\langle y,\beta\rangle) & \triangleq & \\
(\nu y\beta)\,(!\,[\![Q\widehat{\alpha} \dagger \widehat{x}P]\!]_{\mathrm N} \mid \overline{\gamma}\langle y,\beta\rangle) & \triangleq & [\![\widehat{y}(Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma]\!]_{\mathrm N}
\end{array}$$

$(imp\dagger)$: Excluded from $\to_{\mathrm H}$.

$(cut\dagger) : (Q\widehat{\beta} \dagger \widehat{y}R)\widehat{\alpha} \dagger \widehat{x}P \to (Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta} \dagger \widehat{y}(R\widehat{\alpha} \dagger \widehat{x}P)$:

$$\begin{array}{lll}
[\![(Q\widehat{\beta} \dagger \widehat{y}R)\widehat{\alpha} \dagger \widehat{x}P]\!]_{\mathrm N} & \triangleq & \\
(\nu \alpha)\,(!\,(\nu y)\,(!\,[\![Q]\!]_{\mathrm N}[y/\beta] \mid !\,[\![R]\!]_{\mathrm N}) \mid !\,[\![P]\!]_{\mathrm N}[\alpha/x]) & \approx & (4.7(iii)) \\
(\nu y)\,(!\,(\nu \alpha)\,(!\,[\![Q]\!]_{\mathrm N}[y/\beta] \mid !\,[\![P]\!]_{\mathrm N}[\alpha/x]) \mid !\,(\nu \alpha)\,(!\,[\![R]\!]_{\mathrm N} \mid !\,[\![P]\!]_{\mathrm N}[\alpha/x])) & \triangleq & \\
[\![(Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta} \dagger \widehat{y}(R\widehat{\alpha} \dagger \widehat{x}P)]\!]_{\mathrm N} & &
\end{array}$$

Right propagation: $(\dagger cap) : P\widehat{\alpha} \dagger \widehat{x}\langle y{\cdot}\beta\rangle \to \langle y{\cdot}\beta\rangle, y \neq x$:

$$\begin{array}{l}
[\![P\widehat{\alpha} \dagger \widehat{x}\langle y{\cdot}\beta\rangle]\!]_{\mathrm N} \quad \triangleq \quad (\nu \alpha)\,(!\,[\![P]\!]_{\mathrm N} \mid !\,[\![\langle y{\cdot}\beta\rangle]\!]_{\mathrm N}) \ \equiv \\
(\nu \alpha)\,(!\,[\![P]\!]_{\mathrm N}) \mid !\,[\![\langle y{\cdot}\beta\rangle]\!]_{\mathrm N} \ \underset{\sim}{\sqsupset}\ [\![P]\!]_{\mathrm N}
\end{array}$$

$(\dagger exp) : P\widehat{\alpha} \dagger \widehat{x}(\widehat{y}Q\widehat{\beta}{\cdot}\gamma) \to \widehat{y}(P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta}{\cdot}\gamma.$: Like $(exp\text{-}in\dagger)$.

$(\dagger imp\text{-}out), (\dagger imp\text{-}in)$: Excluded from $\to_{\mathrm H}$.

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

$$\begin{array}{lll}
[\![P\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R)]\!]_{\mathrm N} & \triangleq & \\
(\nu x)\,(!\,[\![P]\!]_{\mathrm N}[x/\alpha] \mid !\,(\nu y)\,(!\,[\![Q]\!]_{\mathrm N}[y/\beta] \mid !\,[\![R]\!]_{\mathrm N})) & \approx & (4.7(iii)) \\
(\nu y)\,(!\,(\nu x)\,(!\,[\![P]\!]_{\mathrm N}[x/\alpha] \mid !\,[\![Q]\!]_{\mathrm N}[y/\beta]) \mid !\,(\nu x)\,(!\,[\![P]\!]_{\mathrm N}[x/\alpha] \mid !\,[\![R]\!]_{\mathrm N})) & \triangleq & \\
[\![(P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \dagger \widehat{x}R)]\!]_{\mathrm N} & &
\end{array}$$

Contextual rules: By induction. □

Notice that, in this proof, the only place where reduction plays a role is in the logical rules (exp) and $(exp\text{-}imp)$. All other steps are dealt with by equivalence and replication. Moreover, notice that in the formulation of this result, we remove a replication or remove a larger process in rules $(cap\dagger)$ and $(\dagger cap)$ and restrict the observable behaviour.

The following is thereby an immediate consequence.

Theorem 4.9 *If $P \to_{\mathcal{LK}}{}^{*}\, Q$, and in this reduction sequence infinitely many steps are made using rule $(exp\text{-}imp)$, then $[\![P]\!]_{\mathrm N}$ diverges.*

Proof: From the previous proof it is clear that a synchronisation is possible in the encoding of

$$\begin{array}{lcl}
\llbracket P\widehat{\alpha} \dagger \widehat{x}(\langle x{\cdot}\beta\rangle \widehat{\beta} \dagger \widehat{z}Q)\rrbracket_{\mathrm{N}} & \stackrel{\Delta}{=} & \\
(\nu\alpha)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid !\,(\nu z)\,(!\,\llbracket \langle\alpha{\cdot}z\rangle\rrbracket_{\mathrm{N}} \mid !\,\llbracket Q\rrbracket_{\mathrm{N}})) & \approx & (4.7(iii)) \\
(\nu z)\,(!\,(\nu\alpha)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid !\,\llbracket \langle\alpha{\cdot}z\rangle\rrbracket_{\mathrm{N}}) \mid !\,(\nu\alpha)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid !\,\llbracket Q\rrbracket_{\mathrm{N}})) & \equiv & \alpha \notin fp(Q),\ z \notin fs(P) \\
(\nu z)\,(!\,(\nu\alpha)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid !\,\llbracket \langle\alpha{\cdot}z\rangle\rrbracket_{\mathrm{N}}) \mid !\,\llbracket Q\rrbracket_{\mathrm{N}}) \mid !\,(\nu\alpha)\,!\,\llbracket P\rrbracket_{\mathrm{N}} & \underset{\sim}{\sqsupset} & \\
(\nu z)\,(!\,(\nu\alpha)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid !\,\llbracket \langle\alpha{\cdot}z\rangle\rrbracket_{\mathrm{N}}) \mid !\,\llbracket Q\rrbracket_{\mathrm{N}}) & \stackrel{\Delta}{=} & \\
\llbracket (P\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}z\rangle)\widehat{\beta} \dagger \widehat{z}Q\rrbracket_{\mathrm{N}} & \approx & (4.7(iii)) \\
(\nu\alpha)\,(!\,(\nu z)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid !\,\llbracket Q\rrbracket_{\mathrm{N}}) \mid !\,(\nu z)\,(!\,\llbracket \langle\alpha{\cdot}z\rangle\rrbracket_{\mathrm{N}} \mid !\,\llbracket Q\rrbracket_{\mathrm{N}})) & \equiv & z \notin fs(P),\ \alpha \notin fp(Q) \\
!\,(\nu z)\,!\,\llbracket Q\rrbracket_{\mathrm{N}} \mid (\nu\alpha)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid !\,(\nu z)\,(!\,\llbracket \langle\alpha{\cdot}z\rangle\rrbracket_{\mathrm{N}} \mid !\,\llbracket Q\rrbracket_{\mathrm{N}})) & \underset{\sim}{\sqsupset} & \\
(\nu\alpha)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid !\,(\nu z)\,(!\,\llbracket \langle\alpha{\cdot}z\rangle\rrbracket_{\mathrm{N}} \mid !\,\llbracket Q\rrbracket_{\mathrm{N}})) & \stackrel{\Delta}{=} & \llbracket P\widehat{\alpha} \dagger \widehat{x}(\langle x{\cdot}z\rangle \widehat{\beta} \dagger \widehat{z}Q)\rrbracket_{\mathrm{N}}
\end{array}$$

Figure 2: Translation of the looping reduction

reduction rule (*exp-imp*). □

Snce all reductions in the translation in the cases (*exp*) and (*exp-imp*) satisfy the condition of Proposition 3.6, the above result can be restated as:

Corollary 4.10 *If* $P \to_{\mathrm{H}}^{*} Q$, *then* $\llbracket P\rrbracket_{\mathrm{N}} \underset{\sim}{\sqsupset} \llbracket Q\rrbracket_{\mathrm{N}}$.

Example 4.11 Since all reduction steps in Example 2.9 are steps not involving *import*s, the interpretation of the first and last terms there are related via $\underset{\sim}{\sqsupset}$, and no reduction takes place in the simulation of the $\mathcal{LK}$-reduction, as illustrated in Figure 2. Notice that this shows that there the processes ignored through $\underset{\sim}{\sqsupset}$ do not contribute to the observable behaviour.

Example 4.12 (On completeness) We cannot show that the interpretation is *complete*, since not all reductions in the image of the interpretation correspond to head reduction in $\mathcal{LK}$. Consider the term $(\widehat{z}P\widehat{\delta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{u}(Q\widehat{\tau}\,[u]\,\widehat{x}R)$, and assume that u is not introduced in $Q\widehat{\tau}\,[u]\,\widehat{x}R$. Observe that this term is in $\to_{\mathrm{H}}$-normal form.

However, notice that, since

$$\llbracket (\widehat{z}P\widehat{\delta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{u}(Q\widehat{\tau}\,[u]\,\widehat{x}R)\rrbracket_{\mathrm{N}} \ \stackrel{\Delta}{=}\ (\nu u)\,(!\,(\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid \overline{u}\langle z,\delta\rangle) \mid !\,u(\tau,x).(!\,\llbracket Q\rrbracket_{\mathrm{N}} \mid !\,\llbracket R\rrbracket_{\mathrm{N}}))$$

the translation builds a communication for the top-most *cut* $\gamma \dagger u$, which *can* run:

$$\begin{array}{l}
(\nu u)\,(!\,(\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid \overline{u}\langle z,\delta\rangle) \mid !\,u(\tau,x).(!\,\llbracket Q\rrbracket_{\mathrm{N}} \mid !\,\llbracket R\rrbracket_{\mathrm{N}})) \ \to_{\pi}(u) \\
(\nu u)\,(!\,(\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid \overline{u}\langle z,\delta\rangle) \mid \\
\qquad (\nu z\delta)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid !\,\llbracket Q\rrbracket_{\mathrm{N}}[z/\tau] \mid !\,\llbracket R\rrbracket_{\mathrm{N}}[\delta/w]) \mid \\
\qquad\qquad\qquad !\,u(\tau,x).(!\,\llbracket Q\rrbracket_{\mathrm{N}} \mid !\,\llbracket R\rrbracket_{\mathrm{N}}))
\end{array}$$

This is caused by the fact that, in the π-calculus the 'connections' between γ and u are all established 'individually', rather than all 'in one go' as they are in LK.

So in the translation we can perform synchronisations, whereas the translated term is in normal form, therefore we cannot show a completeness result for head reduction.

5 A semantic translation

In this section, we define a translation from terms in $\mathcal{LK}$ onto processes in $\pi_{\langle\rangle}$ that fully respects reduction in $\mathcal{LK}$, modulo bisimulation, as a variant of the natural translation presented above.

In the approach of $\llbracket\cdot\rrbracket_{\mathrm{N}}$, the *import* $P\widehat{\alpha}\,[x]\,\widehat{y}Q$ is expressed using $x(\alpha,y).(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid !\,\llbracket Q\rrbracket_{\mathrm{N}})$, where the plug α and the socket x become variables that will be replaced by, respectively, an output and input name of the communicating process. We will now modify that definition to make also reduction inside the translation of an *import* possible; for that, we need to revert to a previous version of the natural translation.

The original approach of [7] was, essentially following the translation of $\mathcal{X}$ into $\lambda\mu\tilde{\mu}$, to interpret $P\widehat{\alpha}\,[x]\,\widehat{y}Q$ via

$$x(v,d).((\nu\alpha)\,(!\,[\![P]\!]\,|\,!\,\alpha\rightarrow\overline{v})\,|\,(\nu y)\,(!\,d\rightarrow\overline{y}\,|\,!\,[\![Q]\!]))$$

so, rather than seeing α as an input variable, it sees α as the name of an output channel, and send its output to the input name that will be received in the variable v using the forwarder $\alpha\rightarrow\overline{v}$; similarly, y is seen as an input channel, and the output from the channel which name will arrive in d is redirected to y via $d\rightarrow\overline{y}$. Since output on α might be generated more than once inside $[\![P]\!]$, as well as input might be called for more than once on y inside $[\![Q]\!]$, both forwarders are replicated.

However, using this approach the variables v and d appear *only* in the redirections, not in $[\![P]\!]$ or $[\![Q]\!]$, so these two processes appear unnecessarily under input in the translation. This is what the new translation $[\![\cdot]\!]_{\mathrm{s}}$ fixes: we build what we call a *communication cell* in $x(v,d).(!\,\alpha\rightarrow\overline{v}\,|\,!\,d\rightarrow\overline{y})$, which deals with the redirections of the received *import*'s interface, which we put in parallel with the translations of $[\![P]\!]_{\mathrm{s}}$ and $[\![Q]\!]_{\mathrm{s}}$.

We also choose 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* interpreted terms. This replicated translation is easier to understand, but differs from the natural one in that it does not model reduction via reduction, but via bisimilarity (so does not really constitute an interpretation, but more a semantics),[14] whereas the natural translation truly uses $\pi_{\langle\rangle}$'s reduction in the proofs.

We define:

Definition 5.1 (Semantic translation of $\mathcal{LK}$ into $\pi_{\langle\rangle}$)

$$\begin{array}{rcl}
[\![\langle x{\cdot}\beta\rangle]\!]_{\mathrm{s}} & = & !\,x(w).\overline{\beta}w \\
[\![\widehat{z}P\widehat{\alpha}{\cdot}\beta]\!]_{\mathrm{s}} & = & !\,(\nu z\alpha)\,([\![P]\!]_{\mathrm{s}}\,|\,\overline{\beta}\langle z,\alpha\rangle) \\
[\![P\widehat{\alpha}\,[x]\,\widehat{z}Q]\!]_{\mathrm{s}} & = & !\,(\nu\alpha z)\,([\![P]\!]_{\mathrm{s}}\,|\,x(v,d).(!\,\alpha\rightarrow\overline{v}\,|\,!\,d\rightarrow\overline{z})\,|\,[\![Q]\!]_{\mathrm{s}}) \\
[\![P\widehat{\alpha}\dagger\widehat{z}Q]\!]_{\mathrm{s}} & = & !\,(\nu\alpha z)\,([\![P]\!]_{\mathrm{s}}\,|\,!\,\alpha\rightarrow\overline{z}\,|\,[\![Q]\!]_{\mathrm{s}})
\end{array}$$

Notice that (for technical reasons) we also choose to use the forwarder in the translation of the *cut*, rather than using the renaming mechanism of Definition 4.1.

Remark 5.2 The encoding $[\![\cdot]\!]_{\mathrm{s}}$ generates a flat parallel composition of processes of the shape

$$x(w).\overline{a}\,w \qquad \overline{\alpha}\langle y,\gamma\rangle \qquad x(v,d).(!\,\alpha\rightarrow\overline{v}\,|\,!\,d\rightarrow\overline{y}) \qquad \alpha\rightarrow\overline{x}$$

where all channel names (so not the variables) are coming from the interpreted $\mathcal{LK}$-terms. Synchronisation over these channel names is possible only if generated by the interpretation of *cut*s.

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

Lemma 5.3 $[\![P]\!]_{\mathrm{s}} \approx\, !\,[\![P]\!]_{\mathrm{s}}$.

Proof: Immediate. □

Notice that this lemma implies that we cannot model reduction in $\mathcal{LK}$ via synchronisation; however, as above (Theorem 4.8), we can show a preservation result for this translation modulo

[14] This is comparable to [44], where a similar result is shown for β-equality in the λ-calculus.

weak bisimilarity.

Theorem 5.4 (Operational soundness for $\llbracket\cdot\rrbracket_{\mathrm{s}}$ with respect to $\rightarrow_{\mathcal{LK}}$)
If $P \rightarrow^{*}_{\mathcal{LK}} Q$, *then* $\llbracket P \rrbracket_{\mathrm{s}} \mathrel{\underset{\approx}{\sqsupset}} \llbracket Q \rrbracket_{\mathrm{s}}$.

Proof: By induction on the definition of reduction in $\mathcal{LK}$: again we only show the interesting cases.

$$\begin{array}{lll}
(\mathit{cap}):\ \llbracket \langle y{\cdot}\alpha\rangle \widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle \rrbracket_{\mathrm{s}} & \triangleq & !\,(\nu\alpha x)\,(\llbracket \langle y{\cdot}\alpha\rangle \rrbracket_{\mathrm{s}} \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket \langle x{\cdot}\gamma\rangle \rrbracket_{\mathrm{s}}) \ \triangleq \\
\quad !\,(\nu\alpha x)\,(!\,y(w).\overline{\alpha}w \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, !\,x(w).\overline{\gamma}w) & \approx & !\,y(w).\overline{\gamma}w \ \triangleq\ \llbracket \langle y{\cdot}\gamma\rangle \rrbracket_{\mathrm{s}}
\end{array}$$

$$\begin{array}{lll}
\textit{Logical rules: } (\mathit{exp}):\ \llbracket (\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x{\cdot}\gamma\rangle \rrbracket_{\mathrm{s}} & \triangleq & \\
\quad !\,(\nu\alpha x)\,(\llbracket \widehat{y}P\widehat{\beta}{\cdot}\alpha \rrbracket_{\mathrm{s}} \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket \langle x{\cdot}\gamma\rangle \rrbracket_{\mathrm{s}}) & \triangleq & \\
\quad !\,(\nu\alpha x)\,(!\,(\nu y\beta)\,(\llbracket P \rrbracket_{\mathrm{s}} \,|\, \overline{\alpha}\langle y,\beta\rangle) \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, !\,x(w).\overline{\gamma}w) & \approx & (\alpha,x) \\
\quad !\,(\nu y\beta)\,(\llbracket P \rrbracket_{\mathrm{s}} \,|\, \overline{\gamma}\langle y,\beta\rangle) & \triangleq & \llbracket \widehat{y}P\widehat{\beta}{\cdot}\gamma \rrbracket_{\mathrm{s}}
\end{array}$$

$$\begin{array}{lll}
(\mathit{imp}):\ \llbracket \langle y{\cdot}\alpha\rangle \widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{z}P) \rrbracket_{\mathrm{s}} & \triangleq & \\
\quad !\,(\nu\alpha x)\,(!\,y(w).\overline{\alpha}w \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, !\,(\nu\beta z)\,(\llbracket Q \rrbracket_{\mathrm{s}} \,|\, x(v,d).(!\,\beta{\rightarrow}\overline{v} \,|\, !\,d{\rightarrow}\overline{z}) \,|\, \llbracket P \rrbracket_{\mathrm{s}})) & \approx & (\alpha,x) \\
\quad !\,(\nu\beta z)\,(\llbracket Q \rrbracket_{\mathrm{s}} \,|\, y(v,d).(!\,\beta{\rightarrow}\overline{v} \,|\, !\,d{\rightarrow}\overline{z}) \,|\, \llbracket P \rrbracket_{\mathrm{s}}) & \triangleq & \llbracket Q\widehat{\beta}\,[y]\,\widehat{z}P \rrbracket_{\mathrm{s}}
\end{array}$$

$$\begin{array}{lll}
(\mathit{exp\text{-}imp}):\ \llbracket (\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \rrbracket_{\mathrm{s}} & \triangleq & \\
\quad !\,(\nu\alpha x)\,(!\,(\nu y\beta)\,(\llbracket P \rrbracket_{\mathrm{s}} \,|\, \overline{\alpha}\langle y,\beta\rangle) \,|\, !\,\alpha{\rightarrow}\overline{x} \,| & & \\
\qquad\qquad !\,(\nu\gamma z)\,(\llbracket Q \rrbracket_{\mathrm{s}} \,|\, x(v,d).(!\,\gamma{\rightarrow}\overline{v} \,|\, !\,d{\rightarrow}\overline{z}) \,|\, \llbracket R \rrbracket_{\mathrm{s}})) & \approx & (\alpha,x) \\
\quad (\nu y\beta\gamma z)\,(\llbracket P \rrbracket_{\mathrm{s}} \,|\, \llbracket Q \rrbracket_{\mathrm{s}} \,|\, !\,\gamma{\rightarrow}\overline{y} \,|\, !\,\beta{\rightarrow}\overline{z} \,|\, \llbracket R \rrbracket_{\mathrm{s}}) & \approx & \\
\quad !\,(\nu\gamma y)\,(\llbracket Q \rrbracket_{\mathrm{s}} \,|\, !\,\gamma{\rightarrow}\overline{y} \,|\, !\,(\nu\beta z)\,(\llbracket P \rrbracket_{\mathrm{s}} \,|\, !\,\beta{\rightarrow}\overline{z} \,|\, \llbracket R \rrbracket_{\mathrm{s}})) & \triangleq & \llbracket Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R) \rrbracket_{\mathrm{s}}
\end{array}$$

For the second alternative of this rule, the proof is similar.

$$\begin{array}{lll}
\textit{Left propagation: } (\mathit{cap}\dagger):\ \llbracket \langle y{\cdot}\beta\rangle \widehat{\alpha} \dagger \widehat{x}P \rrbracket_{\mathrm{s}} & \triangleq & \\
\quad !\,(\nu\alpha x)\,(\llbracket \langle y{\cdot}\beta\rangle \rrbracket_{\mathrm{s}} \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket P \rrbracket_{\mathrm{s}}) & \equiv & (\beta \neq \alpha) \\
\quad \llbracket \langle y{\cdot}\beta\rangle \rrbracket_{\mathrm{s}} \,|\, (\nu\alpha x)\,(!\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket P \rrbracket_{\mathrm{s}}) & \underset{\approx}{\sqsupset} & \llbracket \langle y{\cdot}\beta\rangle \rrbracket_{\mathrm{s}}
\end{array}$$

$$\begin{array}{lll}
(\mathit{exp\text{-}out}\dagger):\ \llbracket (\widehat{y}Q\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}P \rrbracket_{\mathrm{s}} & \triangleq & \\
\quad !\,(\nu\alpha x)\,(!\,(\nu y\beta)\,(\llbracket Q \rrbracket_{\mathrm{s}} \,|\, \overline{\alpha}\langle y,\beta\rangle) \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket P \rrbracket_{\mathrm{s}}) & \approx & (5.3) \\
\quad !\,(\nu\alpha x)\,(!\,(\nu y\beta)\,(\llbracket Q \rrbracket_{\mathrm{s}} \,|\, \overline{\alpha}\langle y,\beta\rangle) \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket P \rrbracket_{\mathrm{s}} \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket P \rrbracket_{\mathrm{s}}) & =_{\alpha} & \\
\quad !\,(\nu\gamma x)\,(!\,(\nu y\beta)\,(!\,(\nu\alpha x)\,(\llbracket Q \rrbracket_{\mathrm{s}} \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket P \rrbracket_{\mathrm{s}}) \,|\, \overline{\gamma}\langle y,\beta\rangle) \,|\, !\,\gamma{\rightarrow}\overline{x} \,|\, \llbracket P \rrbracket_{\mathrm{s}}) & \triangleq & \\
\quad \llbracket (\widehat{y}(Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta}{\cdot}\gamma)\widehat{\gamma} \dagger \widehat{x}P \rrbracket_{\mathrm{s}} & &
\end{array}$$

$$\begin{array}{lll}
(\mathit{imp}\dagger):\ \llbracket (Q\widehat{\beta}\,[z]\,\widehat{y}R)\widehat{\alpha} \dagger \widehat{x}P \rrbracket_{\mathrm{s}} & \triangleq & \\
\quad !\,(\nu\alpha x)\,(!\,(\nu\beta y)\,(\llbracket Q \rrbracket_{\mathrm{s}} \,|\, z(v,d).(!\,\beta{\rightarrow}\overline{v} \,|\, !\,d{\rightarrow}\overline{y}) \,|\, \llbracket R \rrbracket_{\mathrm{s}}) \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket P \rrbracket_{\mathrm{s}}) & \approx & (5.3) \\
\quad !\,(\nu\alpha x)\,(!\,(\nu\beta y)\,(\llbracket Q \rrbracket_{\mathrm{s}} \,|\, z(v,d).(!\,\beta{\rightarrow}\overline{v} \,|\, !\,d{\rightarrow}\overline{y}) \,|\, \llbracket R \rrbracket_{\mathrm{s}}) \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket P \rrbracket_{\mathrm{s}} \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket P \rrbracket_{\mathrm{s}}) & =_{\alpha} & \\
\quad !\,(\nu\beta y)\,(!\,(\nu\alpha x)\,(\llbracket Q \rrbracket_{\mathrm{s}} \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket P \rrbracket_{\mathrm{s}}) \,| & & \\
\qquad\qquad z(v,d).(!\,\beta{\rightarrow}\overline{v} \,|\, !\,d{\rightarrow}\overline{y}) \,|\, !\,(\nu\alpha x)\,(\llbracket R \rrbracket_{\mathrm{s}} \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket P \rrbracket_{\mathrm{s}})) & \triangleq & \\
\quad !\,(\nu\beta y)\,(\llbracket Q\widehat{\alpha} \dagger \widehat{x}P \rrbracket_{\mathrm{s}} \,|\, z(v,d).(!\,\beta{\rightarrow}\overline{v} \,|\, !\,d{\rightarrow}\overline{y}) \,|\, \llbracket R\widehat{\alpha} \dagger \widehat{x}P \rrbracket_{\mathrm{s}}) & \triangleq & \\
\quad \llbracket (Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta}\,[z]\,\widehat{y}(R\widehat{\alpha} \dagger \widehat{x}P) \rrbracket_{\mathrm{s}} & &
\end{array}$$

$$\begin{array}{lll}
\textit{Right propagation: } (\dagger\mathit{exp}):\ \llbracket P\widehat{\alpha} \dagger \widehat{x}(\widehat{y}Q\widehat{\beta}{\cdot}\gamma) \rrbracket_{\mathrm{s}} & \triangleq & \\
\quad !\,(\nu\alpha x)\,(\llbracket P \rrbracket_{\mathrm{s}} \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, !\,(\nu y\beta)\,(\llbracket Q \rrbracket_{\mathrm{s}} \,|\, \overline{\gamma}\langle y,\beta\rangle)) & \equiv & \\
\quad !\,(\nu y\beta)\,(!\,(\nu\alpha x)\,(\llbracket P \rrbracket_{\mathrm{s}} \,|\, !\,\alpha{\rightarrow}\overline{x} \,|\, \llbracket Q \rrbracket_{\mathrm{s}}) \,|\, \overline{\gamma}\langle y,\beta\rangle) & \triangleq & \\
\quad !\,(\nu y\beta)\,(\llbracket P\widehat{\alpha} \dagger \widehat{x}Q \rrbracket_{\mathrm{s}} \,|\, \overline{\gamma}\langle y,\beta\rangle) & \triangleq & \llbracket \widehat{y}(P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta}{\cdot}\gamma \rrbracket_{\mathrm{s}}
\end{array}$$

Example 5.5

$$
\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{s}} & \stackrel{\Delta}{=} & \\
!(\nu\gamma u)(\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{s}} \mid !\gamma\rightarrow\overline{u} \mid \llbracket Q\widehat{\tau}\,[u]\,\widehat{w}R\rrbracket_{\mathrm{s}}) & \stackrel{\Delta}{=} & \\
!(\nu\gamma u)(\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{s}} \mid !\gamma\rightarrow\overline{u} \mid !(\nu\tau w)(\llbracket Q\rrbracket_{\mathrm{s}} \mid u(v,d).(!\tau\rightarrow\overline{v} \mid !d\rightarrow\overline{w}) \mid \llbracket R\rrbracket_{\mathrm{s}})) & \approx & (5.3)\\
!(\nu\gamma y)(\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{s}} \mid !\gamma\rightarrow\overline{y} \mid !(\nu\tau w)(!(\nu\gamma u)(\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{s}} \mid !\gamma\rightarrow\overline{u} \mid \llbracket Q\rrbracket_{\mathrm{s}}) \mid & & \\
\qquad\qquad y(v,d).(!\tau\rightarrow\overline{v} \mid !d\rightarrow\overline{w}) \mid !(\nu\gamma u)(\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{s}} \mid !\gamma\rightarrow\overline{u} \mid \llbracket R\rrbracket_{\mathrm{s}}))) & \underset{\sim}{\sqsupset}, \stackrel{\Delta}{=} & (=_{\alpha})\\
!(\nu\gamma y)(\llbracket \widehat{x}P\widehat{\rho}\cdot\gamma\rrbracket_{\mathrm{s}} \mid !\gamma\rightarrow\overline{y} \mid !(\nu\tau w)(!(\nu\gamma u)(!(\nu z\delta)(\llbracket \langle x\cdot\rho\rangle\rrbracket_{\mathrm{s}} \mid \overline{\gamma}\langle z,\delta\rangle) \mid !\gamma\rightarrow\overline{u} \mid & & \\
\qquad\qquad !u(w).\overline{\tau}w) \mid y(v,d).(!\tau\rightarrow\overline{v} \mid !d\rightarrow\overline{w}) \mid \llbracket R\rrbracket_{\mathrm{s}})) & \approx & (\gamma,u)\\
!(\nu\gamma y)(\llbracket \widehat{x}P\widehat{\rho}\cdot\gamma\rrbracket_{\mathrm{s}} \mid !\gamma\rightarrow\overline{y} \mid !(\nu\tau w)(!(\nu z\delta)(\llbracket \langle x\cdot\rho\rangle\rrbracket_{\mathrm{s}} \mid \overline{\tau}\langle z,\delta\rangle) \mid & & \\
\qquad\qquad y(v,d).(!\tau\rightarrow\overline{v} \mid !d\rightarrow\overline{w}) \mid \llbracket R\rrbracket_{\mathrm{s}})) & \stackrel{\Delta}{=} & \\
!(\nu\gamma y)(!(\nu x\rho)(\llbracket \langle x\cdot\rho\rangle\rrbracket_{\mathrm{s}} \mid \overline{\gamma}\langle x,\rho\rangle) \mid !\gamma\rightarrow\overline{y} \mid & & \\
\qquad\qquad !(\nu\tau w)(\llbracket \widehat{z}P\widehat{\delta}\cdot\tau\rrbracket_{\mathrm{s}} \mid y(v,d).(!\tau\rightarrow\overline{v} \mid !d\rightarrow\overline{w}) \mid \llbracket R\rrbracket_{\mathrm{s}})) & \approx & (\gamma,y)\\
!(\nu\gamma y)((\nu x\rho)(\llbracket \langle x\cdot\rho\rangle\rrbracket_{\mathrm{s}} \mid (\nu\tau w)(\llbracket \widehat{z}P\widehat{\delta}\cdot\gamma\rrbracket_{\mathrm{s}} \mid !\tau\rightarrow\overline{x} \mid !\rho\rightarrow\overline{w} \mid !\llbracket R\rrbracket_{\mathrm{s}}))) & \equiv & \\
!(\nu\tau x)(\llbracket \widehat{z}P\widehat{\delta}\cdot\tau\rrbracket_{\mathrm{s}} \mid !\tau\rightarrow\overline{x} \mid !(\nu\rho w)(!x(w).\overline{\rho}w \mid !\rho\rightarrow\overline{w} \mid \llbracket R\rrbracket_{\mathrm{s}})) & \stackrel{\Delta}{=} & \\
\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{s}} & \approx & (5.3)\\
!(\nu\rho w)(!(\nu\tau x)(\llbracket \widehat{z}P\widehat{\delta}\cdot\tau\rrbracket_{\mathrm{s}} \mid !\tau\rightarrow\overline{x} \mid !x(w).\overline{\rho}w) \mid !\rho\rightarrow\overline{w} \mid & & \\
\qquad\qquad !(\nu\tau z)(\llbracket \widehat{z}P\widehat{\delta}\cdot\tau\rrbracket_{\mathrm{s}} \mid !\tau\rightarrow\overline{z} \mid \llbracket R\rrbracket_{\mathrm{s}})) & \underset{\sim}{\sqsupset}, \stackrel{\Delta}{=} & \\
!(\nu\rho w)(!(\nu\tau x)(!(\nu z\delta)(\llbracket P\rrbracket_{\mathrm{s}} \mid \overline{\tau}\langle z,\delta\rangle) \mid !\tau\rightarrow\overline{x} \mid !x(w).\overline{\rho}w) \mid !\rho\rightarrow\overline{w} \mid \llbracket R\rrbracket_{\mathrm{s}}) & \approx & (\tau,x,\rho,w)\\
!(\nu z\delta)(\llbracket P\rrbracket_{\mathrm{s}} \mid \overline{\sigma}\langle z,\delta\rangle) & &
\end{array}
$$

Figure 3: Running the semantic translation of the term of Example 2.5

$$
\begin{array}{lll}
(\dagger\mathit{imp\text{-}out}):\ \llbracket P\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[x]\,\widehat{y}R)\rrbracket_{\mathrm{s}} & \stackrel{\Delta}{=} & \\
\quad !(\nu\alpha x)(\llbracket P\rrbracket_{\mathrm{s}} \mid !\alpha\rightarrow\overline{x} \mid !(\nu\beta y)(\llbracket Q\rrbracket_{\mathrm{s}} \mid x(v,d).(!\beta\rightarrow\overline{v} \mid !d\rightarrow\overline{y}) \mid \llbracket R\rrbracket_{\mathrm{s}})) & \approx & (5.3)\\
\quad !(\nu\alpha x)(\llbracket P\rrbracket_{\mathrm{s}} \mid !\alpha\rightarrow\overline{x} \mid \llbracket P\rrbracket_{\mathrm{s}} \mid !\alpha\rightarrow\overline{x} \mid \llbracket P\rrbracket_{\mathrm{s}} \mid !\alpha\rightarrow\overline{x} \mid & & \\
\qquad\qquad !(\nu\beta y)(\llbracket Q\rrbracket_{\mathrm{s}} \mid x(v,d).(!\beta\rightarrow\overline{v} \mid !d\rightarrow\overline{y}) \mid \llbracket R\rrbracket_{\mathrm{s}})) & =_{\alpha} & \\
\quad !(\nu\alpha z)(\llbracket P\rrbracket_{\mathrm{s}} \mid !\alpha\rightarrow\overline{z} \mid !(\nu\beta y)(!(\nu\alpha x)(\llbracket P\rrbracket_{\mathrm{s}} \mid !\alpha\rightarrow\overline{x} \mid \llbracket Q\rrbracket_{\mathrm{s}}) \mid & & \\
\qquad\qquad z(v,d).(!\beta\rightarrow\overline{v} \mid !d\rightarrow\overline{y}) \mid !(\nu\alpha x)(\llbracket P\rrbracket_{\mathrm{s}} \mid !\alpha\rightarrow\overline{x} \mid \llbracket R\rrbracket_{\mathrm{s}}))) & \stackrel{\Delta}{=} & \\
\quad \llbracket P\widehat{\alpha} \dagger \widehat{z}((P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \dagger \widehat{x}R))\rrbracket_{\mathrm{s}} & &
\end{array}
$$

$$
\begin{array}{lll}
(\dagger\mathit{imp\text{-}in}):\ \llbracket P\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta}\,[z]\,\widehat{y}R)\rrbracket_{\mathrm{s}} & \stackrel{\Delta}{=} & \\
\quad !(\nu\alpha x)(\llbracket P\rrbracket_{\mathrm{s}} \mid !\alpha\rightarrow\overline{x} \mid !(\nu\beta y)(\llbracket Q\rrbracket_{\mathrm{s}} \mid z(v,d).(!\beta\rightarrow\overline{v} \mid !d\rightarrow\overline{y}) \mid \llbracket R\rrbracket_{\mathrm{s}})) & \approx & (5.3)\\
\quad !(\nu\alpha x)(\llbracket P\rrbracket_{\mathrm{s}} \mid !\alpha\rightarrow\overline{x} \mid \llbracket P\rrbracket_{\mathrm{s}} \mid !\alpha\rightarrow\overline{x} \mid !(\nu\beta y)(\llbracket Q\rrbracket_{\mathrm{s}} \mid z(v,d).(!\beta\rightarrow\overline{v} \mid !d\rightarrow\overline{y}) \mid \llbracket R\rrbracket_{\mathrm{s}})) & \approx & \\
\quad !(\nu\beta y)(!(\nu\alpha x)(\llbracket P\rrbracket_{\mathrm{s}} \mid !\alpha\rightarrow\overline{x} \mid \llbracket Q\rrbracket_{\mathrm{s}}) \mid z(v,d).(!\beta\rightarrow\overline{v} \mid !d\rightarrow\overline{y}) \mid !(\nu\alpha x)(\llbracket P\rrbracket_{\mathrm{s}} \mid !\alpha\rightarrow\overline{x} \mid \llbracket R\rrbracket_{\mathrm{s}})) & \stackrel{\Delta}{=} & \\
\quad \llbracket (P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta}\,[z]\,\widehat{y}(P\widehat{\alpha} \dagger \widehat{x}R)\rrbracket_{\mathrm{s}} & &
\end{array}
$$

The contextual rules, as well as the transitive closure, follow by induction. □

Notice that in this proof $\underset{\sim}{\sqsupset}$ is only needed in part $(\mathit{cap}\dagger)$ and $(\dagger\mathit{cap})$, where we eliminate part of a process, otherwise the images of the terms are congruent or bisimilar.

Simulating the reduction of Example 2.5, using the semantic translation, runs as in Figure 3. As suggested by the proof of Theorem 5.4, the $\underset{\sim}{\sqsupset}$ steps correspond to $(\mathit{cap}\dagger)$ and $(\dagger\mathit{cap})$.

This observation leads to:

Theorem 5.6 (Operational completeness for $\llbracket\cdot\rrbracket_{\mathrm{s}}$ with respect to $\rightarrow_{\mathcal{LK}}$) *If* $\llbracket P\rrbracket_{\mathrm{s}} \rightarrow_{\pi} Q$ *then there exists* $P' \in \mathcal{LK}$, R *such that* $Q \rightarrow_{\pi}^{*} R$, $!R \underset{\sim}{\sqsupset} \llbracket P'\rrbracket_{\mathrm{s}}$, *and* $P \rightarrow_{\mathcal{LK}}^{+} P'$.

Proof: From Remark 5.2 we know that the encoding $\llbracket\cdot\rrbracket_{\mathrm{s}}$ generates a flat parallel composition of processes of the shape

$$!\,x(w).\overline{\alpha}\,w \qquad !\,\overline{\alpha}\langle y,\gamma\rangle \qquad !\,x(v,d).(!\,\alpha\to\overline{v} \mid !\,d\to\overline{y}) \qquad !\,\alpha\to\overline{x}$$

where all channel names are coming from the interpreted $\mathcal{LK}$-terms, and synchronisation over these channel names is possible only if generated by the interpretation of *cut*s. By Theorem 5.4, all synchronisations that become possible later correspond to interpreted *cut*s as well. □

As mentioned in Section 2, Gentzen has shown that the innermost reduction strategy on *cut*-elimination for LK is normalising; in the context of $\mathcal{LK}$, this corresponds to showing that the innermost reduction strategy $\to_{\text{I}}$ on $\mathcal{LK}$ is normalising for *typeable* terms.

Proposition 5.7 (GENTZEN'S HAUPTSATZ RESULT FOR $\mathcal{LK}$) *If $P : \Gamma \vdash_{\text{LK}} \Delta$ (so P is typeable), then there exists Q in normal form such that $P \to_{\text{I}}^{*} Q$.*

We will show the equivalent of this result in the setting of the semantic interpretation. We first show:

Lemma 5.8 If P is in normal form, then so is $[\![P]\!]_{\text{S}}$.

Proof: First, notice that in P, the alphabets of sockets and plugs are distinct, and so are the names of input and output channels in $[\![P]\!]_{\text{S}}$. The only exceptions are when P is a cut $Q\widehat{\alpha} \dagger \widehat{z}R$ and $[\![Q\widehat{\alpha} \dagger \widehat{z}R]\!]_{\text{S}} = !\,(\nu\alpha z)\,([\![Q]\!]_{\text{S}} \mid !\,\alpha\to\overline{z} \mid [\![R]\!]_{\text{S}})$ where the forwarder $!\,\alpha\to\overline{z} = \alpha(w).\overline{z}\,w$ is added; notice that here the output α is used as input, and the input z as output. Also in the synchronisation cell inside the interpretation of the import $Q\widehat{\alpha}\,[x]\,\widehat{z}R$, $!\,(\nu\alpha z)\,([\![Q]\!]_{\text{S}} \mid x(v,d).(!\,\alpha\to\overline{v} \mid !\,d\to\overline{z}) \mid [\![R]\!]_{\text{S}})$, this reversal is there. However, since in both cases α and z are restricted, these are not 'reachable' from outside, so can be ignored in this reasoning.

We continue by induction on the structure of terms; notice that P cannot be a cut.

$P = \langle x{\cdot}\alpha\rangle$: Since $[\![\langle x{\cdot}\alpha\rangle]\!]_{\text{S}} = !\,x(w).\overline{\alpha}w$, this is immediate.

$P = \widehat{x}Q\widehat{\alpha}{\cdot}\beta$: $[\![\widehat{x}Q\widehat{\alpha}{\cdot}\beta]\!]_{\text{S}} = !\,(\nu x\alpha)\,([\![Q]\!]_{\text{S}} \mid \overline{\beta}\langle x,\alpha\rangle)$, and by induction, $[\![Q]\!]_{\text{S}}$ is in normal form. Since β is not used inside $[\![Q]\!]_{\text{S}}$ as input, no synchronisation is possible in $!\,(\nu x\alpha)\,([\![Q]\!]_{\text{S}} \mid \overline{\beta}\langle x,\alpha\rangle)$ over β and also $[\![\widehat{x}Q\widehat{\alpha}{\cdot}\beta]\!]_{\text{S}}$ is in normal form.

$P = Q\widehat{\alpha}\,[x]\,\widehat{z}R$: $[\![Q\widehat{\alpha}\,[x]\,\widehat{z}R]\!]_{\text{S}} = !\,(\nu\alpha z)\,([\![Q]\!]_{\text{S}} \mid x(v,d).(!\,\alpha\to\overline{v} \mid !\,d\to\overline{z}) \mid [\![R]\!]_{\text{S}})$, and by induction, $[\![Q]\!]_{\text{S}}$ and $[\![R]\!]_{\text{S}}$ are in normal form. Since x is not an output in either $[\![Q]\!]_{\text{S}}$ or $[\![R]\!]_{\text{S}}$, and input and output names are distinct, no synchronisation is possible between $[\![Q]\!]_{\text{S}}$ and $[\![R]\!]_{\text{S}}$, so also $[\![Q\widehat{\alpha}\,[x]\,\widehat{z}R]\!]_{\text{S}}$ is in normal form. □

Now for termination, we can show:

Theorem 5.9 *If $P \to_{\text{I}}^{*} Q$, and Q is in normal form, then there exists an R in normal form such that $[\![P]\!]_{\text{S}} \approx \mathsf{R}$ and $\mathsf{R} \sqsupseteq_{\sim} [\![Q]\!]_{\text{S}}$.*

Proof: By Lemma 5.8, $[\![Q]\!]_{\text{S}}$ is in normal form. By Theorem 5.4, there exists R such that $[\![P]\!]_{\text{S}} \approx \mathsf{R}$ and $\mathsf{R} \sqsupseteq_{\sim} [\![Q]\!]_{\text{S}}$. By the proof of that theorem, $\sqsupseteq_{\sim}$ is only needed in part $(cap\dagger)$ and $(\dagger cap)$, where we eliminate part of a process, otherwise the images of the terms are congruent or bisimilar. Since now only innermost reduction steps in $P \to_{\text{I}}^{*} Q$ are simulated, $\approx$ deals with communications between processes in normal form, so by Lemma 5.8, whenever $(cap\dagger)$ or $(\dagger cap)$ are simulated, the 'discarded' term corresponds to a process in normal form. So when reaching $[\![Q]\!]_{\text{S}}$, the whole process R is in normal form. □

Combining this with Gentzen's Hauptsatz result, we get:

Theorem 5.10 (PRESERVATION OF TYPEABLE TERMINATION) *Let P be a typeable term, then there exists R in normal form such that $[\![P]\!]_{\text{S}} \approx \mathsf{R}$.*

Proof: If P is typeable, then by Gentzen's result, innermost reduction terminates, so there exists Q in normal form such that $P \rightarrow^*_{\text{I}} Q$; then, by Theorem 5.9, there exists an R in normal form such that $\llbracket P \rrbracket_{\text{S}} \approx R$. $\square$

Notice that we cannot show this result for the natural translation this way, since by Example 4.12 not every term in $\rightarrow_{\text{H}}$ normal form is interpreted by a process in normal form.

Example 5.11 As to the significance of typeability in the previous result, remark that we can encode, via $\ulcorner\cdot\rrbracket^{\lambda}_{\cdot}$ (an encoding that maps λ-terms to LK-terms; for details, see [9]), the (non-terminating) λ-term $(\lambda x.xx)(\lambda x.xx) = \Delta\Delta$ in $\mathcal{LK}$.

Using that definition, we have

$$\begin{array}{rcl}
\ulcorner xx \rrbracket^{\lambda}_{\sigma} & \triangleq & \ulcorner x \rrbracket^{\lambda}_{\gamma}\widehat{\gamma} \dagger \widehat{y}(\ulcorner x \rrbracket^{\lambda}_{\delta}\widehat{\delta}\,[y]\,\widehat{v}\langle v{\cdot}\beta\rangle) \\
& \triangleq & \langle x{\cdot}\gamma\rangle\,\widehat{\gamma} \dagger \widehat{y}(\langle x{\cdot}\delta\rangle\,\widehat{\delta}\,[y]\,\widehat{v}\langle v{\cdot}\beta\rangle) \\
& \rightarrow_{\text{I}} & \langle x{\cdot}\delta\rangle\,\widehat{\delta}\,[x]\,\widehat{y}\langle y{\cdot}\beta\rangle
\end{array}$$

We write this last term as $\mathbf{xx}_{\beta}$; notice that there is a redex inside $\ulcorner xx \rrbracket^{\lambda}_{\beta}$.

Now $\ulcorner \Delta\Delta \rrbracket^{\lambda}_{\beta}$ reduces innermost as follows:

$$\begin{array}{lll}
\ulcorner \Delta\Delta \rrbracket^{\lambda}_{\beta} \quad \triangleq \quad \ulcorner \lambda x.xx \rrbracket^{\lambda}_{\gamma}\widehat{\gamma} \dagger \widehat{z}(\ulcorner \lambda x.xx \rrbracket^{\lambda}_{\gamma}\widehat{\gamma}\,[z]\,\widehat{y}\langle y{\cdot}\beta\rangle) & \triangleq & \\
(\widehat{x}\ulcorner xx \rrbracket^{\lambda}_{\alpha}\widehat{\alpha}{\cdot}\delta)\,\widehat{\delta} \dagger \widehat{z}((\widehat{x}\ulcorner xx \rrbracket^{\lambda}_{\alpha}\widehat{\alpha}{\cdot}\gamma)\,\widehat{\gamma}\,[z]\,\widehat{y}\langle y{\cdot}\beta\rangle) & \rightarrow_{\text{I}} & (2\times) \\
(\widehat{x}\,\mathbf{xx}_{\alpha}\;\widehat{\alpha}{\cdot}\delta)\,\widehat{\delta} \dagger \widehat{z}((\widehat{x}\,\mathbf{xx}_{\sigma}\;\widehat{\sigma}{\cdot}\gamma)\,\widehat{\gamma}\,[z]\,\widehat{y}\langle y{\cdot}\beta\rangle) & \rightarrow_{\text{I}} & \\
(\widehat{x}\,\mathbf{xx}_{\sigma}\;\widehat{\sigma}{\cdot}\gamma)\,\widehat{\gamma} \dagger \widehat{x}(\mathbf{xx}_{\alpha}\;\widehat{\alpha} \dagger \widehat{y}\langle y{\cdot}\beta\rangle) & \triangleq & \\
(\widehat{x}\,\mathbf{xx}_{\sigma}\;\widehat{\sigma}{\cdot}\gamma)\,\widehat{\gamma} \dagger \widehat{x}((\langle x{\cdot}\delta\rangle\,\widehat{\delta}\,[x]\,\widehat{y}\langle y{\cdot}\alpha\rangle)\,\widehat{\alpha} \dagger \widehat{y}\langle y{\cdot}\beta\rangle) & \rightarrow_{\text{I}} & \\
(\widehat{x}\,\mathbf{xx}_{\sigma}\;\widehat{\sigma}{\cdot}\gamma)\,\widehat{\gamma} \dagger \widehat{x}((\langle x{\cdot}\delta\rangle\,\widehat{\alpha} \dagger \widehat{y}\langle y{\cdot}\beta\rangle)\,\widehat{\delta}\,[x]\,\widehat{y}(\langle y{\cdot}\alpha\rangle\,\widehat{\alpha} \dagger \widehat{y}\langle y{\cdot}\beta\rangle)) & \rightarrow_{\text{I}} & (2\times) \\
(\widehat{x}\,\mathbf{xx}_{\sigma}\;\widehat{\sigma}{\cdot}\gamma)\,\widehat{\gamma} \dagger \widehat{x}\;\mathbf{xx}_{\beta} & \triangleq & \\
(\widehat{x}\,\mathbf{xx}_{\sigma}\;\widehat{\sigma}{\cdot}\gamma)\,\widehat{\gamma} \dagger \widehat{x}(\langle x{\cdot}\delta\rangle\,\widehat{\delta}\,[x]\,\widehat{y}\langle y{\cdot}\beta\rangle) & \rightarrow_{\text{I}} & \\
(\widehat{x}\,\mathbf{xx}_{\sigma}\;\widehat{\sigma}{\cdot}\gamma)\,\widehat{\gamma} \dagger \widehat{z}((((\widehat{x}\,\mathbf{xx}_{\sigma}\;\widehat{\sigma}{\cdot}\gamma)\,\widehat{\gamma} \dagger \widehat{x}\langle x{\cdot}\delta\rangle))\,\widehat{\delta}\;[z] & & \\
\qquad\qquad\qquad\qquad \widehat{y}((\widehat{x}\,\mathbf{xx}_{\sigma}\;\widehat{\sigma}{\cdot}\gamma)\,\widehat{\gamma} \dagger \widehat{x}\langle y{\cdot}\beta\rangle)) & \rightarrow_{\text{I}} & (2\times) \\
(\widehat{x}\,\mathbf{xx}_{\sigma}\;\widehat{\sigma}{\cdot}\gamma)\,\widehat{\gamma} \dagger \widehat{z}((\widehat{x}\,\mathbf{xx}_{\sigma}\;\widehat{\sigma}{\cdot}\delta)\,\widehat{\delta}\,[z]\,\widehat{y}\langle y{\cdot}\beta\rangle) & &
\end{array}$$

So the term $\Omega = (\widehat{x}\,\mathbf{xx}_{\sigma}\;\widehat{\sigma}{\cdot}\gamma)\,\widehat{\gamma} \dagger \widehat{z}((\widehat{x}\,\mathbf{xx}_{\sigma}\;\widehat{\sigma}{\cdot}\delta)\,\widehat{\delta}\,[z]\,\widehat{y}\langle y{\cdot}\beta\rangle)$ reduces innermost to itself, and all its reducts contain at least one *cut*. By the proof of Theorem 5.4, each cut of the shape $\llbracket (\widehat{y}P\widehat{\beta}{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\gamma}\,[x]\,\widehat{z}R) \rrbracket_{\text{S}}$ gets interpreted by a process in which a synchronisation is possible; therefore in particular, each innermost reduction sequence (there are many, since rule (*exp-imp*) has two alternatives) from $\llbracket \Omega \rrbracket_{\text{S}}$ contains infinitely many processes that are not in normal form. Of course Ω is not typeable.

6 Type assignment

In this section, we discuss a notion of type assignment $\vdash_{\pi}$ for processes in $\pi_{\langle\rangle}$, as first presented in [7], that describes the '*input-output interface*' of a process. Typeability of a process is expressed via the ternary relation

$$P : \Gamma \vdash^{io}_{\pi} \Delta$$

(so almost identical to that for $\mathcal{LK}$), where P is a process that is said to be the *witness* to the judgement $\Gamma \vdash \Delta$, the left context Γ contains pairs of channel names and types for all the input channels of P, and the right context Δ for its output channels; since in P a channel can be used for both, it can appear in both contexts. Our system thereby gives an abstract functional translation of processes by stating the connectability of a process via giving the names of the available (connectable) channels and their types.

We aim to show that all typeable terms in LK translate to typeable processes, i.e.: if P is a witness to a judgement (in $\vdash_{\text{LK}}$), then its translations via $[\![\cdot]\!]_{\text{N}}$ and $[\![\cdot]\!]_{\text{S}}$ are as well. Together with the preservation results we have shown above, this implies that we can not only interpret reduction in LK through synchronisation (similarly to what was done, for example, in [42] for the lazy λ-calculus), but actually show that the processes we create through our interpretations accurately represent the *actual proofs*, so our synchronisations correctly model *cut*-elimination, and transform a proof into a proof.

We first define a notion of type assignment that is more traditionally stated, in that if assigns a collection of types to (free) channel names of a process and derives statements of the shape $\Psi \vdash_\pi P$. We will show a subject reduction result for this system, and then define a second notion, that separates input and outputs, and derives statements of the shape $P : \Gamma \vdash^{io}_\pi \Delta$, and show that then that judgement is equivalent to $\Gamma, \Delta \vdash_\pi P$.

Definition 6.1 (Functional type assignment for $\pi_{\langle\rangle}$) *i*) The types A and contexts Ψ we consider for $\pi_{\langle\rangle}$ are defined like those of Definition 1.3, generalised to names.

ii) Functional type assignment for $\pi_{\langle\rangle}$ is defined by the following deduction system:

$$(0) : \frac{}{\vdash 0} \qquad (!) : \frac{\Psi \vdash P}{\Psi \vdash\, !P} \qquad (out) : \frac{}{a{:}A, b{:}A \vdash \overline{a}\, b}\ (a \neq b)$$

$$(\nu) : \frac{\Psi, a{:}A \vdash P}{\Psi \vdash (\nu a)\, P}\ (a \notin \Psi) \qquad (\langle\rangle\text{-}out) : \frac{}{a{:}A{\to}B, b{:}A, c{:}B \vdash \overline{a}\langle b,c\rangle}\ (a,b,c \textit{ different})$$

$$(|) : \frac{\Psi \vdash P_1 \quad \cdots \quad \Psi \vdash P_n}{\Psi \vdash P_1 \mid \cdots \mid P_n} \qquad (in) : \frac{\Psi, x{:}A \vdash P}{\Psi, a{:}A \vdash a(x).P}\ (a \neq x,\ x \notin \Psi)$$

$$(Wk) : \frac{\Psi \vdash P}{\Psi' \vdash P}\ (\Psi' \supseteq \Psi) \qquad (let) : \frac{\Psi, x{:}A, y{:}B \vdash P}{\Psi, z{:}A{\to}B \vdash let\ \langle x,y\rangle = z\ in\ P}\ (x,y,z \notin \Psi \textit{ and different})$$

iii) As usual, we write $\Psi \vdash_\pi P$ if there exists a derivation using these rules that has the expression $\Psi \vdash P$ in the conclusion.

Notice that the rule (let) is only applicable when z is not free in P.

The notion $\vdash_\pi$ is a true type assignment system which does not (directly) relate back to LK.[15] For example, rule (0) makes 0 a witness to an unprovable judgement, and rules $(|)$ and $(!)$ do not change the contexts, so do not correspond to any rule in the logic, not even to a $\lambda\mu$-style [43] activation step. Moreover, rule (ν) just removes a formula, and rule $(\langle\rangle\text{-}out)$ is clearly not an instance of an axiom in LK; notice that that rule does not directly correspond to the logical rule $({\Rightarrow}R)$, as that $(\langle\rangle\text{-}in)$ 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 6.8.

This notion is novel 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:

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

Moreover, we use arrow types, expressing that processes can be seen as functions from an input to an output channel. In order to be able to interpret proofs in LK, types in our system need not be decorated with channel information, but will express functionality instead.

Example 6.2 We can derive

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

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

so the following rule is derivable:

$$(\langle\rangle\text{-}in):\quad \dfrac{\Psi, y{:}B, x{:}A \vdash P}{\Psi, a{:}A{\to}B \vdash a(x,y).P}\quad \begin{array}{l}(y,a \notin \Delta,\, x \notin \Psi\\ \quad a,x,y \text{ different})\end{array}$$

We can show a consistency result for this notion of type assignment, for which we first need to prove a substitution lemma.

Proposition 6.3 (Substitution) *If* $\Psi, x{:}A \vdash_\pi P$*, then* $\Psi, b{:}A \vdash_\pi P[b/x]$ *for* b *fresh or* $b{:}A \in \Psi$.

Proof: Easy. □

Theorem 6.4 (Subject reduction for $\vdash_\pi$) *If* $\Psi \vdash_\pi P$ *and* $P \to_\pi^* Q$*, then* $\Psi \vdash_\pi Q$.

Proof: We only deal with the base cases.

$\overline{a}\,b \mid a(z).P \to P[b/z]$: The derivation for $\overline{a}\,b \mid a(z).P$ is shaped like:

$$\dfrac{\dfrac{}{a{:}A, b{:}A \vdash_\pi \overline{a}\,b}\;(out) \qquad \dfrac{\dfrac{\boxed{}}{\Psi, z{:}A \vdash_\pi P}}{\Psi, a{:}A \vdash_\pi a(z).P}\;(in)}{\Psi, a{:}A{\to}B \vdash_\pi \overline{a}\,b \mid a(z).P}\;(\mid)$$

By Lemma 6.3, we can derive

$$\dfrac{\boxed{}}{\Psi, b{:}A \vdash_\pi P[b/z]}$$

$\overline{a}\langle b,c\rangle \mid a(z).P \to P[\langle a,b\rangle/z]$: By induction on the structure of P; we only show the two base cases in which $\langle a,b\rangle/z$ is well defined:

$P = \overline{d}\,z$: Then $P[\langle b,c\rangle/z] = \overline{d}\langle b,c\rangle$. The derivation for $\overline{a}\langle b,c\rangle \mid a(z).\overline{d}\,z$ is shaped like:

$$\dfrac{\dfrac{}{b{:}A, a{:}A{\to}B, c{:}B \vdash_\pi \overline{a}\langle b,c\rangle}\;(\langle\rangle\text{-}out) \qquad \dfrac{\dfrac{}{d{:}A{\to}B, z{:}A{\to}B \vdash_\pi \overline{d}\,z}\;(out)}{d{:}A{\to}B, a{:}A{\to}B \vdash_\pi a(z).\overline{d}\,z}\;(in)}{b{:}A, a{:}A{\to}B, c{:}B, d{:}A{\to}B \vdash_\pi \overline{a}\langle b,c\rangle \mid a(z).\overline{d}\,z}\;(\mid)$$

We can construct:

$$\dfrac{\dfrac{}{b{:}A, d{:}A{\to}B, c{:}B \vdash_\pi \overline{d}\langle b,c\rangle}\;(\langle\rangle\text{-}out)}{b{:}A, a{:}A{\to}B, c{:}B, d{:}A{\to}B \vdash_\pi \overline{d}\langle b,c\rangle}\;(Wk)$$

$P = \mathsf{let}\,\langle x,y\rangle = z\ \mathsf{in}\ Q$: Then $P[\langle b,c\rangle/z] = Q[b/x, c/y]$. The derivation for the process $\overline{a}\langle b,c\rangle \mid a(z).\mathsf{let}\,\langle x,y\rangle = z\ \mathsf{in}\ Q$ is shaped like:

$$\dfrac{\dfrac{}{b{:}A, a{:}A{\to}B, c{:}B \vdash_\pi \overline{a}\langle b,c\rangle}\;(\langle\rangle\text{-}out) \qquad \dfrac{\dfrac{\dfrac{\boxed{}}{\Psi, x{:}A, y{:}B \vdash_\pi Q}}{\Psi, z{:}A{\to}B \vdash_\pi \mathsf{let}\,\langle x,y\rangle = z\ \mathsf{in}\ Q}\;(\mathsf{let})}{\Psi, a{:}A{\to}B \vdash_\pi a(z).\mathsf{let}\,\langle x,y\rangle = z\ \mathsf{in}\ Q}\;(in)}{\Psi, b{:}A, a{:}A{\to}B, c{:}B \vdash_\pi \overline{a}\langle b,c\rangle \mid a(z).\mathsf{let}\,\langle x,y\rangle = z\ \mathsf{in}\ Q}\;(\mid)$$

By Lemma 6.3, we can derive

$$\dfrac{\dfrac{\boxed{}}{\Psi,b{:}A,c{:}B \vdash_\pi Q[b/x,c/y]}}{\Psi,b{:}A,a{:}A{\to}B,c{:}B \vdash_\pi Q[b/x,c/y]}\;(Wk)$$

□

We will now come to our type preservation results. Note that we could aim to show that if $P : \Gamma \vdash_{\text{LK}} \Delta$, then $\Gamma,\Delta \vdash_\pi [\![P]\!]$; however, in this result the latter sequent does not respect the left and right context. So rather, we will introduce the notion of type assignment we defined in [7]; it was formulated slightly differently, more in the style of LK, by separating input and output channels in the derivable sequents.

Definition 6.5 (IMPLICATIVE IO TYPE ASSIGNMENT FOR $\pi_{\langle\rangle}$ [7]) *i*) The types and contexts we consider for $\pi_{\langle\rangle}$ are defined like those of Definition 1.3, generalised to names, but allowing both Roman and Greek names on *both* sides of the turnstile.

ii) Type assignment for $\pi_{\langle\rangle}$ is defined by the following sequent system:[16]

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

$$(\nu) : \frac{P : \Gamma, a{:}A \vdash a{:}A, \Delta}{(\nu a)\,P : \Gamma \vdash \Delta}\;(a \notin \Gamma,\Delta) \qquad (out) : \frac{}{\overline{a}\,b : b{:}A \vdash a{:}A, b{:}A}\;(a \neq b)$$

$$(|) : \frac{P_1 : \Gamma \vdash \Delta \quad \cdots \quad P_n : \Gamma \vdash \Delta}{P_1 \mid \cdots \mid P_n : \Gamma \vdash \Delta} \qquad (in) : \frac{P : \Gamma, x{:}A \vdash x{:}A, \Delta}{a(x).P : \Gamma, a{:}A \vdash \Delta}$$

$$(Wk) : \frac{P : \Gamma \vdash \Delta}{P : \Gamma' \vdash \Delta'}\;(\Gamma' \supseteq \Gamma, \Delta' \supseteq \Delta) \qquad (let) : \frac{P : \Gamma, y{:}B \vdash x{:}A, \Delta}{let\,\langle x,y\rangle = z\ in\ P : \Gamma, z{:}A{\to}B \vdash \Delta}\;\begin{array}{l}(y,z \notin \Delta;\\ \;x,z \notin \Gamma)\end{array}$$

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

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

As in Example 6.2 the following rule is derivable:

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

Since in the synchronisation step $\overline{a}\langle b,c\rangle \mid a(x,y).Q \to_\pi Q[b/x,c/y]$, in $\overline{a}\langle b,c\rangle$ the name b is used for input, and c for output whereas their role is reversed in $Q[b/x,c/y]$, we cannot show a straightforward witness reduction result for $\vdash^{io}_\pi$ as we did for $\vdash_\pi$ in Theorem 6.4. We can, however, show that there exists a strong link between the two notions of type assignment. First we need to formally define the notions of input and output names for processes.

Definition 6.6 For a process P we define the set of its *input names* $I(P)$ and that of its *output names* $O(P)$ as follows:

[16] Notice that we have taken the liberty to reuse the names of the rules.

$I(\mathsf{0})$	$=\emptyset$	$O(\mathsf{0})$	$=\emptyset$
$I(P \mid Q)$	$= I(P) \cup I(Q)$	$O(P \mid Q)$	$= O(P) \cup O(Q)$
$I(!\,P)$	$= I(P)$	$O(!\,P)$	$= O(P)$
$I((\nu a)\,P)$	$= I(P) \setminus \{a\}$	$O((\nu a)\,P)$	$= O(P) \setminus \{a\}$
$I(a(x).P)$	$= I(P) \cup \{a\} \setminus \{x\}$	$O(a(x).P)$	$= O(P) \setminus \{x\}$
$I(\overline{a}\,b)$	$= \{b\}$	$O(\overline{a}\,b)$	$= \{a,b\}$
$I(\overline{a}\langle b,c\rangle)$	$= \{b\}$	$O(\overline{a}\langle b,c\rangle)$	$= \{a,c\}$
$I(\mathit{let}\,\langle x,y\rangle = z \ \mathit{in}\ P)$	$= I(P) \cup \{z\} \setminus \{y\}$	$O(\mathit{let}\,\langle x,y\rangle = z \ \mathit{in}\ P)$	$= O(P) \setminus \{x\}$

We can now show a direct relation between $\vdash_{\pi}$ and $\vdash^{io}_{\pi}$.

Lemma 6.7 *i) If $\Psi \vdash_{\pi} P$, and $\Gamma = \{a{:}A \in \Psi \mid a \in I(P)\}$, $\Delta = \{a{:}A \in \Psi \mid a \in O(P)\}$, then $P : \Gamma \vdash^{io}_{\pi} \Delta$.*
ii) If $P : \Gamma \vdash^{io}_{\pi} \Delta$, then $\Gamma, \Delta \vdash_{\pi} P$.

Proof: By induction on the structure of derivations. We only give the more interesting parts. Remember that, by Definition 1.3, we allow $a{:}B$ to occur in Γ when we write $\Gamma, a{:}A$, but then $A = B$, and that $\Gamma, a{:}A = \Gamma \backslash x \cup \{x{:}A\}$.

- (ν): Then $P = (\nu d)\,Q$. For (i), then $\Psi, d{:}A \vdash_{\pi} Q$ for some A, with $d \notin \Psi$. Take $\Gamma = \{a{:}A \in \Psi, d{:}A \mid a \in I(Q)\}$ and $\Delta = \{a{:}A \in \Psi, d{:}A \mid a \in O(Q)\}$, then by induction $Q : \Gamma \vdash^{io}_{\pi} \Delta$; a priori, we do not know if $d{:}A$ occurs in either context, but in any case we can construct

$$\dfrac{\dfrac{\begin{array}{c}\boxed{\quad\quad}\\ Q : \Gamma \vdash \Delta\end{array}}{Q : \Gamma, d{:}A \vdash d{:}A, \Delta}\,(\mathit{Wk})}{(\nu d)\,Q : \Gamma\backslash d \vdash \Delta\backslash d}\,(\nu)$$

  Notice that $\Gamma\backslash d = \{a{:}A \in \Psi \mid a \in I((\nu d)\,Q)\}$ and $\Delta\backslash d = \{a{:}A \in \Psi \mid a \in O((\nu a)\,Q)\}$. For (ii), then $Q : \Gamma, d{:}A \vdash^{io}_{\pi} d{:}A, \Delta$, for some A. By induction, $\Gamma, d{:}A, \Delta \vdash_{\pi} Q$, and by (ν), also $\Gamma, \Delta \vdash_{\pi} (\nu d)\,Q$.

- (in): Then $P = d(x).Q$. For (i), then $\Psi = \Psi', d{:}A$, for some A, and $\Psi', x{:}A \vdash_{\pi} Q$, with $x \notin \Psi'$. Take $\Gamma = \{a{:}A \in \Psi', x{:}A \mid a \in I(Q)\}$ and $\Delta = \{a{:}A \in \Psi', x{:}A \mid a \in O(Q)\}$, then by induction $Q : \Gamma \vdash^{io}_{\pi} \Delta$ and we can construct

$$\dfrac{\dfrac{\begin{array}{c}\boxed{\quad\quad}\\ Q : \Gamma \vdash^{io}_{\pi} \Delta\end{array}}{Q : \Gamma, x{:}A \vdash^{io}_{\pi} x{:}A, \Delta}\,(\mathit{Wk})}{d(x).Q : \Gamma\backslash x, d{:}A \vdash^{io}_{\pi} d{:}A, \Delta\backslash x}\,(\mathit{in})$$

  Notice that $\Gamma\backslash x, d{:}A = \{a{:}A \in \Psi', d{:}A \mid a \in I(d(x).Q)\}$ and $\Delta\backslash x, d{:}A = \{a{:}A \in \Psi', d{:}A \mid a \in O(d(x).Q)\}$. For (ii), then $Q : \Gamma, d{:}A \vdash^{io}_{\pi} d{:}A, \Delta$, for some A. By induction, $\Gamma, d{:}A, \Delta \vdash_{\pi} Q$, and by (ν), also $\Gamma, \Delta \vdash_{\pi} d(x).Q$.

- (let): Then $P = \mathit{let}\,\langle x,y\rangle = z \ \mathit{in}\ Q$. For (i), then $\Psi = \Psi', z{:}A \rightarrow B$, for some A, and $\Psi', x{:}A, y{:}B \vdash_{\pi} Q$, with $x, y, z \notin \Psi'$. Take $\Gamma = \{a{:}A \in \Psi', x{:}A, y{:}B \mid a \in I(Q)\}$ and $\Delta = \{a{:}A \in \Psi', x{:}A, y{:}B \mid a \in O(Q)\}$, then by induction $Q : \Gamma \vdash^{io}_{\pi} \Delta$ and we can construct

$$\dfrac{\dfrac{\begin{array}{c}\boxed{\quad\quad}\\ Q : \Gamma \vdash^{io}_{\pi} \Delta\end{array}}{Q : \Gamma, y{:}B \vdash^{io}_{\pi} x{:}A, \Delta}\,(\mathit{Wk})}{\mathit{let}\,\langle x,y\rangle = z \ \mathit{in}\ Q : \Gamma\backslash y, z{:}A \rightarrow B \vdash^{io}_{\pi} \Delta\backslash x}\,(\mathit{let})$$

  Notice that $\Gamma\backslash y, z{:}A \rightarrow B = \{a{:}A \in \Psi', z{:}A \rightarrow B \mid a \in I(\mathit{let}\,\langle x,y\rangle = z \ \mathit{in}\ Q)\}$ and $\Delta\backslash x = \{a{:}A \in$

$\Psi', z{:}A \rightarrow B \mid a \in O\,(\mathit{let}\,\langle x,y\rangle = z \;\mathit{in}\; Q)\,\}$.

For (*ii*), then $Q : \Gamma, y{:}A \vdash_{\pi}^{io} x{:}A, \Delta$, for some A, with $y \notin \Gamma$, $x \notin \Delta$, and $z \notin \Gamma,\Delta$. By induction, $\Gamma, y{:}A, x{:}B, \Delta \vdash_{\pi} Q$, and $\Gamma, \Delta, z{:}A \rightarrow B \vdash_{\pi} \mathit{let}\,\langle x,y\rangle = z \;\mathit{in}\; Q$ follows by applying (let). $\square$

This result implies that we can see $\vdash_{\pi}^{io}$ as a 'sugared' version of $\vdash_{\pi}$, where for readability channel names are separated in input and output names.

We now come to the main soundness result for our notion of type assignment for $\pi_{\langle\rangle}$. The following theorem shows that the natural translation $\llbracket\cdot\rrbracket_{\mathrm{N}}$ preserves assignable types.

Theorem 6.8 (Type preservation for the natural interpretation) *$P : \Gamma \vdash_{\mathbf{LK}} \Delta$ if and only if $\llbracket P \rrbracket_{\mathrm{N}} : \Gamma \vdash_{\pi}^{io} \Delta$.*

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

$\langle x{\cdot}\alpha\rangle$: Then $\llbracket \langle x{\cdot}\alpha\rangle \rrbracket_{\mathrm{N}} = x(w).\overline{\alpha}\,w$, and there exists A such that the $\mathcal{LK}$-derivation is shaped like:

$$\dfrac{}{\langle x{\cdot}\alpha\rangle : \Gamma', x{:}A \vdash \alpha{:}A, \Delta'}\;(\mathit{cap})$$

Since $\llbracket \langle x{\cdot}\alpha\rangle \rrbracket_{\mathrm{N}} = x(w).\overline{\alpha}\,w$, the structure of the derivation for $\llbracket \langle x{\cdot}\alpha\rangle \rrbracket_{\mathrm{N}} : \Gamma \vdash_{\pi}^{io} \Delta$ must include an (in), preceded by (out). By that last rule, there exists A such that $\overline{\alpha}\,w : w{:}A \vdash_{\pi}^{io} \alpha{:}A, w{:}A$, yielding

$$\dfrac{\dfrac{\dfrac{}{\overline{\alpha}\,w : w{:}A \vdash_{\pi}^{io} \alpha{:}A, w{:}A}\;(\mathit{out})}{x(w).\overline{\alpha}\,w : x{:}A \vdash_{\pi}^{io} \alpha{:}A}\;(\mathit{in})}{x(w).\overline{\alpha}\,w : \Gamma', x{:}A \vdash_{\pi}^{io} \alpha{:}A, \Delta'}\;(\mathit{Wk})$$

adding weakening to generalise the result.

$\widehat{x}P\widehat{\alpha}{\cdot}\beta$: Since $\llbracket \widehat{x}P\widehat{\alpha}{\cdot}\beta \rrbracket_{\mathrm{N}} = (\nu x\alpha)\,(!\,\llbracket P \rrbracket_{\mathrm{N}} \mid \overline{\beta}\langle x,\alpha\rangle)$, by induction we have $P : \Gamma \vdash \Delta$ if and only if $\llbracket P \rrbracket_{\mathrm{N}} : \Gamma \vdash_{\pi}^{io} \Delta$. Then there exists A and B such that $\mathcal{LK}$-derivation is shaped like:

$$\dfrac{\dfrac{\boxed{}}{P : \Gamma', x{:}A \vdash \alpha{:}B, \Delta'}}{\widehat{x}P\widehat{\alpha}{\cdot}\beta : \Gamma' \vdash \beta{:}A{\rightarrow}B, \Delta'}\;(\mathit{exp})$$

The structure of the derivation for $(\nu x\alpha)\,(!\,\llbracket P \rrbracket_{\mathrm{N}} \mid \overline{\beta}\langle x,\alpha\rangle) : \Gamma \vdash_{\pi}^{io} \Delta$ must include instances of (ν), $(\mid)$, $(!)$, and $(\langle\rangle\text{-}\mathit{out})$. By this last rule there exists A and B such that $\overline{\beta}\langle x,\alpha\rangle : x{:}A \vdash_{\pi}^{io} \alpha{:}B, \beta{:}A{\rightarrow}B$, yielding

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\boxed{}}{\llbracket P \rrbracket_{\mathrm{N}} : \Gamma', x{:}A \vdash_{\pi}^{io} \alpha{:}B, \Delta'}}{!\,\llbracket P \rrbracket_{\mathrm{N}} : \Gamma', x{:}A \vdash_{\pi}^{io} \alpha{:}B, \Delta'}\;(!) \qquad \dfrac{}{\overline{\beta}\langle x,\alpha\rangle : x{:}A \vdash_{\pi}^{io} \alpha{:}B, \beta{:}A{\rightarrow}B}\;(\langle\rangle\text{-}\mathit{out})}{!\,\llbracket P \rrbracket_{\mathrm{N}} \mid \overline{\beta}\langle x,\alpha\rangle : \Gamma', x{:}A \vdash_{\pi}^{io} \alpha{:}B, \beta{:}A{\rightarrow}B, \Delta'}\;(\mid)}{(\nu\alpha)\,(!\,\llbracket P \rrbracket_{\mathrm{N}} \mid \overline{\beta}\langle x,\alpha\rangle) : \Gamma', x{:}A \vdash_{\pi}^{io} \beta{:}A{\rightarrow}B, \Delta'}\;(\nu)}{(\nu x\alpha)\,(!\,\llbracket P \rrbracket_{\mathrm{N}} \mid \overline{\beta}\langle x,\alpha\rangle) : \Gamma' \vdash_{\pi}^{io} \beta{:}A{\rightarrow}B, \Delta'}\;(\nu)$$

$P\widehat{\alpha}\,[y]\,\widehat{x}Q$: Since $\llbracket P\widehat{\alpha}\,[y]\,\widehat{x}Q \rrbracket_{\mathrm{N}} = y(\alpha,x).(!\,\llbracket P \rrbracket_{\mathrm{N}} \mid !\,\llbracket Q \rrbracket_{\mathrm{N}})$, by induction we have $P : \Gamma \vdash \Delta$ if and only if $\llbracket P \rrbracket_{\mathrm{N}} : \Gamma \vdash_{\pi}^{io} \Delta$. and $Q : \Gamma \vdash \Delta$ if and only if $\llbracket Q \rrbracket_{\mathrm{N}} : \Gamma \vdash_{\pi}^{io} \Delta$.

There exists A and B such that the $\mathcal{LK}$-derivation is shaped like:

$$\dfrac{\dfrac{\boxed{}}{P : \Gamma' \vdash \alpha{:}A, \Delta'} \qquad \dfrac{\boxed{}}{Q : \Gamma', x{:}B \vdash \Delta'}}{P\widehat{\alpha}\,[y]\,\widehat{x}Q : \Gamma', y{:}A{\rightarrow}B \vdash \Delta'}\;(\mathit{imp})$$

The structure of the derivation for $y(\alpha,x).(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid !\,\llbracket Q\rrbracket_{\mathrm{N}}) : \Gamma \vdash^{io}_{\pi} \Delta$ must include instances of $(\langle\rangle\textit{-in})$, $(\mid)$, and $(!\,)$. By rule $(\langle\rangle\textit{-in})$ there are A and B such that $y{:}A \to B$, yielding

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ \llbracket P\rrbracket_{\mathrm{N}} : \Gamma' \vdash^{io}_{\pi} \alpha{:}A,\Delta'\end{array}}{!\,\llbracket P\rrbracket_{\mathrm{N}} : \Gamma' \vdash^{io}_{\pi} \alpha{:}A,\Delta'}\,(!) \qquad \dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ \llbracket Q\rrbracket_{\mathrm{N}} : \Gamma',x{:}B \vdash^{io}_{\pi} \Delta'\end{array}}{!\,\llbracket Q\rrbracket_{\mathrm{N}} : \Gamma',x{:}B \vdash^{io}_{\pi} \Delta'}\,(!)}{!\,\llbracket P\rrbracket_{\mathrm{N}} \mid !\,\llbracket Q\rrbracket_{\mathrm{N}} : \Gamma',x{:}B \vdash^{io}_{\pi} \alpha{:}A,\Delta'}\,(\mid)}{y(\alpha,x).(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid !\,\llbracket Q\rrbracket_{\mathrm{N}}) : \Gamma',y{:}A{\to}B \vdash^{io}_{\pi} \Delta'}\,(\langle\rangle\textit{-in})$$

$P\widehat{\alpha} \dagger \widehat{x}Q$: Since $\llbracket P\widehat{\alpha} \dagger \widehat{x}Q\rrbracket_{\mathrm{N}} = (\nu x)\,(!\,\llbracket P\rrbracket_{\mathrm{N}}[x/\alpha] \mid !\,\llbracket Q\rrbracket_{\mathrm{N}})$, by induction we have $P : \Gamma \vdash \Delta$ if and only if $\llbracket P\rrbracket_{\mathrm{N}} : \Gamma \vdash^{io}_{\pi} \Delta$ and $Q : \Gamma \vdash \Delta$ if and only if $\llbracket Q\rrbracket_{\mathrm{N}} : \Gamma \vdash^{io}_{\pi} \Delta$.
Then the $\mathcal{LK}$-derivation is shaped like:

$$\dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ P : \Gamma' \vdash \alpha{:}A,\Delta'\end{array} \qquad \begin{array}{c}\boxed{\qquad\qquad}\\ Q : \Gamma',x{:}A \vdash \Delta'\end{array}}{P\widehat{\alpha} \dagger \widehat{x}Q : \Gamma' \vdash \Delta'}\,(\textit{cut})$$

Since x does not occur in P, by Lemma 6.3 also for $\llbracket P\rrbracket_{\mathrm{N}}[x/\alpha] : \Gamma' \vdash^{io}_{\pi} x{:}A,\Delta'$. The structure of the derivation for $(\nu x)\,(!\,\llbracket P\rrbracket_{\mathrm{N}}[x/\alpha] \mid !\,\llbracket Q\rrbracket_{\mathrm{N}}) : \Gamma \vdash^{io}_{\pi} \Delta$ must include instances of (ν), $(\mid)$, and $(!\,)$, yielding:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ \llbracket P\rrbracket_{\mathrm{N}}[x/\alpha] : \Gamma' \vdash^{io}_{\pi} x{:}A,\Delta'\end{array}}{!\,\llbracket P\rrbracket_{\mathrm{N}}[x/\alpha] : \Gamma' \vdash^{io}_{\pi} x{:}A,\Delta'}\,(!) \qquad \dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ \llbracket Q\rrbracket_{\mathrm{N}} : \Gamma',x{:}A \vdash^{io}_{\pi} \Delta'\end{array}}{!\,\llbracket Q\rrbracket_{\mathrm{N}} : \Gamma',x{:}A \vdash^{io}_{\pi} \Delta'}\,(!)}{!\,\llbracket P\rrbracket_{\mathrm{N}}[x/\alpha] \mid !\,\llbracket Q\rrbracket_{\mathrm{N}} : \Gamma',x{:}A \vdash^{io}_{\pi} x{:}A,\Delta'}\,(\mid)}{(\nu x)\,(!\,\llbracket P\rrbracket_{\mathrm{N}}[x/\alpha] \mid !\,\llbracket Q\rrbracket_{\mathrm{N}}) : \Gamma' \vdash^{io}_{\pi} \Delta'}\,(\nu)$$

□

As a corollary, through Lemma 6.7 we immediately obtain $P : \Gamma \vdash_{\mathrm{LK}} \Delta \Rightarrow \Gamma,\Delta \vdash_{\pi} \llbracket P\rrbracket_{\mathrm{N}}$.
We can also show that the semantic translation $\llbracket\cdot\rrbracket_{\mathrm{S}}$ preserves assignable types.

Theorem 6.9 (Type preservation for the semantic interpretation) *$P : \Gamma \vdash_{\mathrm{LK}} \Delta$ if and only if $\llbracket P\rrbracket_{\mathrm{S}} : \Gamma \vdash^{io}_{\pi} \Delta$.*

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

$\langle x{\cdot}\alpha\rangle$, $\widehat{x}P\widehat{\alpha}{\cdot}\beta$: These cases are almost identical to those for Theorem 6.8, but for the additional use of rule $(!)$.

$P\widehat{\alpha}\,[y]\,\widehat{x}Q$: Since $\llbracket P\widehat{\alpha}\,[y]\,\widehat{x}Q\rrbracket_{\mathrm{S}} = !\,(\nu\alpha x)\,(\llbracket P\rrbracket_{\mathrm{S}} \mid y(v,d).(!\,\alpha{\to}\overline{v} \mid !\,d{\to}\overline{x}) \mid \llbracket Q\rrbracket_{\mathrm{S}})$, by induction we have $P : \Gamma \vdash \Delta$ if and only if $\llbracket P\rrbracket_{\mathrm{S}} : \Gamma \vdash^{io}_{\pi} \Delta$. and $Q : \Gamma \vdash \Delta$ if and only if $\llbracket Q\rrbracket_{\mathrm{S}} : \Gamma \vdash^{io}_{\pi} \Delta$. There exists A and B such that the $\mathcal{LK}$-derivation is shaped like:

$$\dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ P : \Gamma' \vdash \alpha{:}A,\Delta'\end{array} \qquad \begin{array}{c}\boxed{\qquad\qquad}\\ Q : \Gamma',x{:}B \vdash \Delta'\end{array}}{P\widehat{\alpha}\,[y]\,\widehat{x}Q : \Gamma',y{:}A{\to}B \vdash \Delta'}\,(\textit{imp})$$

The structure of the derivation for $!\,(\nu\alpha x)\,(\llbracket P\rrbracket_{\mathrm{S}} \mid y(v,d).(!\,\alpha{\to}\overline{v} \mid !\,d{\to}\overline{x}) \mid \llbracket Q\rrbracket_{\mathrm{S}}) : \Gamma \vdash^{io}_{\pi} \Delta$ must include instances of (ν), $(\langle\rangle\textit{-in})$, $(\mid)$, $(\textit{in})$, $(\textit{out})$, and $(!\,)$. By rule $(\langle\rangle\textit{-in})$ there are A and B such that $y{:}A \to B$, yielding

$$\dfrac{\dfrac{\boxed{}}{\llbracket P \rrbracket_{\mathrm{s}} : \Gamma' \vdash^{io}_{\pi} \alpha{:}A,\Delta'} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{v}\,w : w{:}A \vdash^{io}_{\pi} v{:}A,w{:}A}\,(out)}{\alpha\rightarrow\overline{v} : \alpha{:}A \vdash^{io}_{\pi} v{:}A}\,(in)}{!\,\alpha\rightarrow\overline{v} : \alpha{:}A \vdash^{io}_{\pi} v{:}A}\,(!) \quad \dfrac{\dfrac{\dfrac{}{\overline{x}\,w : w{:}B \vdash^{io}_{\pi} x{:}B,w{:}B}\,(out)}{d\rightarrow\overline{x} : d{:}B \vdash^{io}_{\pi} x{:}B}\,(in)}{!\,d\rightarrow\overline{x} : d{:}B \vdash^{io}_{\pi} x{:}B}\,(!)}{!\,\alpha\rightarrow\overline{v} \mid !\,d\rightarrow\overline{x} : \alpha{:}A,d{:}B \vdash^{io}_{\pi} x{:}B,v{:}A}\,(\mid)}{y(v,d).(!\,\alpha\rightarrow\overline{v} \mid !\,d\rightarrow\overline{x}) : \alpha{:}A,y{:}A\rightarrow B \vdash^{io}_{\pi} x{:}B}\,(\langle\rangle\text{-}in) \quad \dfrac{\boxed{}}{\llbracket Q \rrbracket_{\mathrm{s}} : \Gamma',x{:}B \vdash^{io}_{\pi} \Delta'}}{\dfrac{\llbracket P \rrbracket_{\mathrm{s}} \mid y(v,d).(!\,\alpha\rightarrow\overline{v} \mid !\,d\rightarrow\overline{x}) \mid \llbracket Q \rrbracket_{\mathrm{s}} : \Gamma',x{:}B,\alpha{:}A,y{:}A\rightarrow B \vdash^{io}_{\pi} x{:}B,\alpha{:}A,\Delta'}{\dfrac{(\nu x)\,(\llbracket P \rrbracket_{\mathrm{s}} \mid y(v,d).(!\,\alpha\rightarrow\overline{v} \mid !\,d\rightarrow\overline{x}) \mid \llbracket Q \rrbracket_{\mathrm{s}}) : \Gamma',\alpha{:}A,y{:}A\rightarrow B \vdash^{io}_{\pi} \alpha{:}A,\Delta'}{\dfrac{(\nu\alpha x)\,(\llbracket P \rrbracket_{\mathrm{s}} \mid y(v,d).(!\,\alpha\rightarrow\overline{v} \mid !\,d\rightarrow\overline{x}) \mid \llbracket Q \rrbracket_{\mathrm{s}}) : \Gamma',y{:}A\rightarrow B \vdash^{io}_{\pi} \Delta'}{!\,(\nu\alpha x)\,(\llbracket P \rrbracket_{\mathrm{s}} \mid y(v,d).(!\,\alpha\rightarrow\overline{v} \mid !\,d\rightarrow\overline{x}) \mid \llbracket Q \rrbracket_{\mathrm{s}}) : \Gamma',y{:}A\rightarrow B \vdash^{io}_{\pi} \Delta'}\,(!)}\,(\nu)}\,(\nu)}\,(\mid)$$

$P\widehat{\alpha} \dagger \widehat{x}Q$: Since $\llbracket P\widehat{\alpha} \dagger \widehat{x}Q \rrbracket_{\mathrm{N}} = !\,(\nu\alpha x)\,(\llbracket P \rrbracket_{\mathrm{s}} \mid !\,\alpha\rightarrow\overline{x} \mid \llbracket Q \rrbracket_{\mathrm{s}})$, by induction we have $P : \Gamma \vdash \Delta$ if and only if $\llbracket P \rrbracket_{\mathrm{s}} : \Gamma \vdash^{io}_{\pi} \Delta$ and $Q : \Gamma \vdash \Delta$ if and only if $\llbracket Q \rrbracket_{\mathrm{s}} : \Gamma \vdash^{io}_{\pi} \Delta$. Then the $\mathcal{LK}$-derivation is shaped like:

$$\dfrac{\dfrac{\boxed{}}{P : \Gamma' \vdash \alpha{:}A,\Delta'} \quad \dfrac{\boxed{}}{Q : \Gamma',x{:}A \vdash \Delta'}}{P\widehat{\alpha} \dagger \widehat{x}Q : \Gamma' \vdash \Delta'}\,(cut)$$

The structure of the derivation for $(\nu x)\,(!\,\llbracket P \rrbracket_{\mathrm{N}}[x/\alpha] \mid !\,\llbracket Q \rrbracket_{\mathrm{N}}) : \Gamma \vdash^{io}_{\pi} \Delta$ must include instances of (ν), $(\mid)$, and $(!\,)$, yielding:

$$\dfrac{\dfrac{\boxed{}}{\llbracket P \rrbracket_{\mathrm{s}} : \Gamma' \vdash^{io}_{\pi} \alpha{:}A,\Delta'} \quad \dfrac{\dfrac{\dfrac{}{\overline{x}\,w : w{:}A \vdash^{io}_{\pi} x{:}A,w{:}A}\,(out)}{\alpha\rightarrow\overline{x} : \alpha{:}A \vdash^{io}_{\pi} x{:}A}\,(in)}{!\,\alpha\rightarrow\overline{x} : \alpha{:}A \vdash^{io}_{\pi} x{:}A}\,(!) \quad \dfrac{\boxed{}}{\llbracket Q \rrbracket_{\mathrm{s}} : \Gamma',x{:}A \vdash^{io}_{\pi} \Delta'}}{\dfrac{\llbracket P \rrbracket_{\mathrm{s}} \mid !\,\alpha\rightarrow\overline{x} \mid \llbracket Q \rrbracket_{\mathrm{s}} : \Gamma',\alpha{:}A,x{:}A \vdash^{io}_{\pi} \alpha{:}A,x{:}A,\Delta'}{\dfrac{(\nu x)\,(\llbracket P \rrbracket_{\mathrm{s}} \mid !\,\alpha\rightarrow\overline{x} \mid \llbracket Q \rrbracket_{\mathrm{s}}) : \Gamma',\alpha{:}A \vdash^{io}_{\pi} \alpha{:}A,\Delta'}{\dfrac{(\nu\alpha x)\,(\llbracket P \rrbracket_{\mathrm{s}} \mid !\,\alpha\rightarrow\overline{x} \mid \llbracket Q \rrbracket_{\mathrm{s}}) : \Gamma' \vdash^{io}_{\pi} \Delta'}{!\,(\nu\alpha x)\,(\llbracket P \rrbracket_{\mathrm{s}} \mid !\,\alpha\rightarrow\overline{x} \mid \llbracket Q \rrbracket_{\mathrm{s}}) : \Gamma' \vdash^{io}_{\pi} \Delta'}\,(!)}\,(\nu)}\,(\nu)}\,(\mid)$$

and $\llbracket P\widehat{\alpha} \dagger \widehat{x}Q \rrbracket_{\mathrm{s}} = !\,(\nu\alpha x)\,(\llbracket P \rrbracket_{\mathrm{s}} \mid !\,\alpha\rightarrow\overline{x} \mid \llbracket Q \rrbracket_{\mathrm{s}})$. □

The following is a direct result of the last two theorems:

Proposition 6.10 (Simulation of cut-elimination) *Assume $P \rightarrow_{\mathcal{LK}} Q$, then :*

i) If $\llbracket P \rrbracket_{\mathrm{N}} : \Gamma \vdash^{io}_{\pi} \Delta$, then $\llbracket Q \rrbracket_{\mathrm{N}} : \Gamma \vdash^{io}_{\pi} \Delta$.

ii) If $\llbracket P \rrbracket_{\mathrm{s}} : \Gamma \vdash^{io}_{\pi} \Delta$, then $\llbracket Q \rrbracket_{\mathrm{s}} : \Gamma \vdash^{io}_{\pi} \Delta$.

Proof: By Theorem 2.6, 6.8, and 6.9. □

Although we have lost subject reduction as a general property for $\vdash^{io}_{\pi}$, we can show that in $\vdash^{io}_{\pi}$ it is preserved for reductions in the image of our translations, for which we first show a contraction result.

Lemma 6.11 (Contraction) *i) If a does not occur in Q, $a \neq b$, and $(\nu b)\,\overline{a}\,b \mid a(x).Q : \Gamma,a{:}C \vdash^{io}_{\pi} a{:}C,\Delta$, then $(\nu b)\,(Q[b/x]) : \Gamma \vdash^{io}_{\pi} \Delta$.*

ii) If a does not occur in P, $a \neq e$, and $(\nu bc)\,(P \mid \overline{a}\langle b,c\rangle) \mid a(x).\overline{e}\,x : \Gamma,a{:}C \vdash^{io}_{\pi} a{:}C,\Delta$, then $(\nu bc)\,(P \mid \overline{e}\langle b,c\rangle) : \Gamma \vdash^{io}_{\pi} \Delta$.

iii) If a does not occur in P and Q and $(\nu bc)\,(P \mid \overline{a}\langle b,c\rangle) \mid a(x,y).Q : \Gamma,a{:}C \vdash^{io}_{\pi} a{:}C,\Delta$, then $(\nu bc)\,(P \mid Q[b/x,c/y]) : \Gamma \vdash^{io}_{\pi} \Delta$.

*iv) If a does not occur in P and $(\nu bc)\,(P \mid \overline{a}\langle b,c\rangle) \mid a(v,d).(\,!\,x \rightarrow \overline{v} \mid !\,d \rightarrow \overline{y}) : \Gamma, a{:}C \vdash^{io}_{\pi} a{:}C, \Delta$, then $(\nu bc)\,(P \mid !\,x \rightarrow \overline{b} \mid !\,c \rightarrow \overline{y})$
$\Gamma \vdash^{io}_{\pi} \Delta$.*

Proof: *i*) Then the derivation is shaped like:

$$\dfrac{\dfrac{\dfrac{}{\overline{a}\,b : b{:}A \vdash^{io}_{\pi} a{:}A, b{:}A}\,(out)}{(\nu b)\,\overline{a}\,b : \vdash^{io}_{\pi} a{:}A}\,(\nu) \qquad \dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ Q : \Gamma', x{:}A \vdash^{io}_{\pi} x{:}A, \Delta'\end{array}}{a(x).Q : \Gamma', a{:}A \vdash^{io}_{\pi} \Delta'}\,(in)}{(\nu b)\,\overline{a}\,b \mid a(x).Q : \Gamma', a{:}A \vdash^{io}_{\pi} a{:}A, \Delta'}\,(\mid)$$

Since b is restricted, it does not occur in Q, by Lemma 6.3 also $Q[b/x] : \Gamma, b{:}A \vdash^{io}_{\pi} b{:}A, \Delta$, and we can construct:

$$\dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ Q[b/x] : \Gamma', b{:}A \vdash^{io}_{\pi} b{:}A, \Delta'\end{array}}{(\nu b)\,Q[b/x] : \Gamma' \vdash^{io}_{\pi} \Delta'}\,(\nu)$$

ii) Then the derivation is shaped like:

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\boxed{\qquad}\\ P : \Gamma' \vdash^{io}_{\pi} \Delta'\end{array} \qquad \dfrac{}{\overline{a}\langle b,c\rangle : b{:}A \vdash^{io}_{\pi} a{:}A\to B, c{:}B}\,(\langle\rangle\text{-}out)}{P \mid \overline{a}\langle b,c\rangle : \Gamma', b{:}A \vdash^{io}_{\pi} a{:}A\to B, c{:}B, \Delta'}\,(\mid)}{(\nu c)\,(P \mid \overline{a}\langle b,c\rangle) : \Gamma', b{:}A \vdash^{io}_{\pi} a{:}A\to B, \Delta'}\,(\nu)}{(\nu bc)\,(P \mid \overline{a}\langle b,c\rangle) : \Gamma' \vdash^{io}_{\pi} a{:}A\to B, \Delta'}\,(\nu) \qquad \dfrac{\dfrac{}{\overline{e}\,x : x{:}A\to B \vdash^{io}_{\pi} e{:}A\to B, x{:}A\to B}\,(out)}{a(x).\overline{e}\,x : a{:}A\to B \vdash^{io}_{\pi} e{:}A\to B}\,(in)}{(\nu bc)\,(P \mid \overline{a}\langle b,c\rangle) \mid a(x).\overline{e}\,x : \Gamma', a{:}A\to B \vdash^{io}_{\pi} a{:}A\to B, e{:}A\to B, \Delta'}\,(\mid)$$

Since b and c are restricted, they are different from e and we can construct:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\boxed{\qquad}\\ P : \Gamma' \vdash^{io}_{\pi} \Delta'\end{array} \qquad \dfrac{}{\overline{e}\langle b,c\rangle : b{:}A \vdash^{io}_{\pi} e{:}A\to B, c{:}B}\,(\langle\rangle\text{-}out)}{P \mid \overline{e}\langle b,c\rangle : \Gamma', b{:}A \vdash^{io}_{\pi} e{:}A\to B, c{:}B, \Delta'}\,(\mid)}{(\nu c)\,(P \mid \overline{e}\langle b,c\rangle) : \Gamma', b{:}A \vdash^{io}_{\pi} e{:}A\to B, \Delta'}\,(\nu)}{(\nu bc)\,(P \mid \overline{e}\langle b,c\rangle) : \Gamma' \vdash^{io}_{\pi} e{:}A\to B, \Delta'}\,(\nu)$$

iii) The derivation is of the shape

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\boxed{\qquad}\\ P : \Gamma \vdash^{io}_{\pi} \Delta\end{array} \qquad \dfrac{}{\overline{a}\langle b,c\rangle : \Gamma, b{:}A \vdash^{io}_{\pi} a{:}A\to B, c{:}B, \Delta}\,(\langle\rangle\text{-}out)}{P \mid \overline{a}\langle b,c\rangle : \Gamma, b{:}A \vdash^{io}_{\pi} a{:}A\to B, \Delta}\,(\mid)}{(\nu c)\,(P \mid \overline{a}\langle b,c\rangle) : \Gamma, b{:}A \vdash^{io}_{\pi} a{:}A\to B, \Delta}\,(\nu)}{(\nu bc)\,(P \mid \overline{a}\langle b,c\rangle) : \Gamma \vdash^{io}_{\pi} a{:}A\to B, \Delta}\,(\nu) \qquad \dfrac{\begin{array}{c}\boxed{\qquad\qquad}\\ Q : \Gamma, y{:}B \vdash^{io}_{\pi} x{:}A, \Delta\end{array}}{a(x,y).Q : \Gamma, a{:}A\to B \vdash^{io}_{\pi} \Delta}\,(\langle\rangle\text{-}in)}{(\nu bc)\,(P \mid \overline{a}\langle b,c\rangle) \mid a(x,y).Q : \Gamma, a{:}A\to B \vdash^{io}_{\pi} a{:}A\to B, \Delta}\,(\mid)$$

Since b and c are restricted, they do not occur in Q, so by Lemma 6.3 also $Q[b/x,c/y]$: $\Gamma, c{:}B \vdash^{io}_{\pi} b{:}A, \Delta$ and we can construct:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\boxed{\qquad}\\ P : \Gamma \vdash^{io}_{\pi} \Delta\end{array} \qquad \begin{array}{c}\boxed{\qquad\qquad}\\ Q[b/x,c/y] : \Gamma, c{:}B \vdash^{io}_{\pi} b{:}A, \Delta\end{array}}{P \mid Q[b/x,c/y] : \Gamma, c{:}B \vdash^{io}_{\pi} b{:}A, \Delta}\,(\mid)}{(\nu c)\,(P \mid Q[b/x,c/y]) : \Gamma \vdash^{io}_{\pi} b{:}A, \Delta}\,(\nu)}{(\nu bc)\,(P \mid Q[b/x,c/y]) : \Gamma \vdash^{io}_{\pi} \Delta}\,(\nu)$$

iv) Then the derivation is of the shape

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{\bigtriangledown}{P : \Gamma' \vdash_{\pi}^{io} \Delta'}
\quad
\dfrac{}{\overline{a}\langle b,c\rangle : b{:}A \vdash_{\pi}^{io} a{:}A{\rightarrow}B, c{:}B}\,(\langle\rangle\text{-}\mathit{out})
}{P \mid \overline{a}\langle b,c\rangle : \Gamma', b{:}A \vdash_{\pi}^{io} a{:}A{\rightarrow}B, \Delta'}\,(\mid)
}{(\nu c)\,(P \mid \overline{a}\langle b,c\rangle) : \Gamma', b{:}A \vdash_{\pi}^{io} a{:}A{\rightarrow}B, \Delta'}\,(\nu)
}{(\nu bc)\,(P \mid \overline{a}\langle b,c\rangle) : \Gamma' \vdash_{\pi}^{io} a{:}A{\rightarrow}B, \Delta'}\,(\nu)
\quad
\dfrac{
\dfrac{
\begin{array}{c}
\dfrac{\dfrac{\dfrac{}{\overline{v}\,w : w{:}A \vdash_{\pi}^{io} v{:}A, w{:}A}\,(\mathit{out})}{x{\rightarrow}\overline{v} : x{:}A \vdash_{\pi}^{io} v{:}A}\,(\mathit{in})}{!\,x{\rightarrow}\overline{v} : x{:}A \vdash_{\pi}^{io} v{:}A}\,(!) \\ \vdots
\end{array}
\quad
\dfrac{\dfrac{\dfrac{}{\overline{y}\,w : w{:}B \vdash_{\pi}^{io} y{:}B, w{:}B}\,(\mathit{out})}{d{\rightarrow}\overline{y} : d{:}B \vdash_{\pi}^{io} y{:}B}\,(\mathit{in})}{!\,d{\rightarrow}\overline{y} : d{:}B \vdash_{\pi}^{io} y{:}B}\,(!)
}{!\,x{\rightarrow}\overline{v} \mid !\,d{\rightarrow}\overline{y} : x{:}A, d{:}B \vdash_{\pi}^{io} y{:}B, v{:}A}\,(\mid)
}{a(v,d).(!\,x{\rightarrow}\overline{v} \mid !\,d{\rightarrow}\overline{y}) : x{:}A, a{:}A{\rightarrow}B \vdash_{\pi}^{io} y{:}B}\,(\langle\rangle\text{-}\mathit{in})
}{(\nu bc)\,(P \mid \overline{a}\langle b,c\rangle) \mid a(v,d).(!\,x{\rightarrow}\overline{v} \mid !\,d{\rightarrow}\overline{y}) : \Gamma', x{:}A, a{:}A{\rightarrow}B \vdash_{\pi}^{io} a{:}A{\rightarrow}B, y{:}B, \Delta'}\,(\mid)
$$

Since b and c are restricted, they are different from x and y and we can construct:

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{\bigtriangledown}{P : \Gamma' \vdash_{\pi}^{io} \Delta'}
\quad
\dfrac{\dfrac{\dfrac{}{\overline{b}\,w : w{:}A \vdash_{\pi}^{io} b{:}A, w{:}A}\,(\mathit{out})}{x{\rightarrow}\overline{b} : x{:}A \vdash_{\pi}^{io} b{:}A}\,(\mathit{in})}{!\,x{\rightarrow}\overline{b} : x{:}A \vdash_{\pi}^{io} b{:}A}\,(!)
\quad
\dfrac{\dfrac{\dfrac{}{\overline{y}\,w : w{:}B \vdash_{\pi}^{io} y{:}B, w{:}B}\,(\mathit{out})}{c{\rightarrow}\overline{y} : c{:}B \vdash_{\pi}^{io} y{:}B}\,(\mathit{in})}{!\,c{\rightarrow}\overline{y} : c{:}B \vdash_{\pi}^{io} y{:}B}\,(!)
}{P \mid !\,x{\rightarrow}\overline{b} \mid !\,c{\rightarrow}\overline{y} : \Gamma', x{:}A, c{:}B \vdash_{\pi}^{io} b{:}A, y{:}B, \Delta'}\,(\mid)
}{(\nu c)\,(P \mid !\,x{\rightarrow}\overline{b} \mid !\,c{\rightarrow}\overline{y}) : \Gamma', x{:}A \vdash_{\pi}^{io} b{:}A, y{:}B, \Delta'}\,(\nu)
}{(\nu bc)\,(P \mid !\,x{\rightarrow}\overline{b} \mid !\,c{\rightarrow}\overline{y}) : \Gamma', x{:}A \vdash_{\pi}^{io} y{:}B, \Delta'}\,(\nu)
\qquad \Box
$$

Using this result, we can also show a witness reduction result:

Theorem 6.12 *i) If* $[\![P]\!]_{\mathrm{N}} : \Gamma \vdash_{\pi}^{io} \Delta$*, and* $[\![P]\!]_{\mathrm{N}} \rightarrow_{\pi}^{*} Q$*, then* $Q : \Gamma \vdash_{\pi}^{io} \Delta$*.*
ii) If $[\![P]\!]_{\mathrm{S}} : \Gamma \vdash_{\pi}^{io} \Delta$*, and* $[\![P]\!]_{\mathrm{S}} \rightarrow_{\pi}^{*} Q$*, then* $Q : \Gamma \vdash_{\pi}^{io} \Delta$*.*

Proof: By Remark 4.4 and 5.2, and Lemma 6.11. $\Box$

7 Expressing Negation

In this section we will look at the logical connective $\neg$, how it is dealt with within $\mathcal{LK}$,[17] and how to interpret it in the π-calculus. Together with the treatment of implication it is then possible to also express all other first-order logical connectives, but we will not deal with those explicitly here.

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

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

To extend the Curry-Howard isomorphism of $\mathcal{LK}$ also to negation, 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 *i*) We extend $\mathcal{LK}$'s syntax with the following constructs:

[17] An alternative way of treating negation is to add the type $\perp$, but in arrow types only on the right-hand side, and let *export* and *import* deal with it, but this would need separate constructs to introduce $\perp$ on either the right or the left; we feel our present approach is more clear.

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

ii) We extend the set of types by

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

(as usual, $\neg A \rightarrow B$ stands for $(\neg A)\rightarrow B$).

iii) We add the type assignment rules:

$$(\textit{r-inv}) : \frac{P : \Gamma, x{:}A \vdash \Delta}{\widehat{x}P{\cdot}\alpha : \Gamma \vdash \alpha{:}\neg A, \Delta} \qquad\qquad (\textit{l-inv}) : \frac{P : \Gamma \vdash \alpha{:}A, \Delta}{x{\cdot}P\widehat{\alpha} : \Gamma, x{:}\neg A \vdash \Delta}$$

Example 7.3 We can inhabit $\neg\neg A \rightarrow A$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle y{\cdot}\alpha\rangle : y{:}A \vdash \alpha{:}A}\;(\textit{cap})}{\widehat{y}\langle y{\cdot}\alpha\rangle{\cdot}\gamma : \;\vdash \gamma{:}\neg A, \alpha{:}A}\;(\textit{r-inv})}{x{\cdot}(\widehat{y}\langle y{\cdot}\alpha\rangle{\cdot}\gamma)\widehat{\gamma} : x{:}\neg\neg A \vdash \alpha{:}A}\;(\textit{l-inv})}{\widehat{x}\,(x{\cdot}(\widehat{y}\langle y{\cdot}\alpha\rangle{\cdot}\gamma)\widehat{\gamma})\widehat{\alpha}{\cdot}\beta : \;\vdash \beta{:}\neg\neg A \rightarrow A}\;(\textit{exp})$$

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

Definition 7.4 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.

The logical reduction rule for negation is (with β and x introduced):

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

We add the propagation rules:

$$\begin{array}{lcll} (y{\cdot}Q\widehat{\beta})\widehat{\alpha} \dagger \widehat{x}P & \rightarrow & y{\cdot}(Q\widehat{\alpha} \dagger \widehat{x}P)\widehat{\beta} & \\ (\widehat{y}Q{\cdot}\beta)\widehat{\alpha} \dagger \widehat{x}P & \rightarrow & \widehat{y}(Q\widehat{\alpha} \dagger \widehat{x}P){\cdot}\beta & (\alpha \neq \beta) \\ (\widehat{y}Q{\cdot}\alpha)\widehat{\alpha} \dagger \widehat{x}P & \rightarrow & (\widehat{y}(Q\widehat{\alpha} \dagger \widehat{x}P){\cdot}\beta)\widehat{\beta} \dagger \widehat{x}P & (\beta\,\textit{fresh}, \alpha\;\textit{not introduced}) \\ & & & \\ P\widehat{\alpha} \dagger \widehat{x}\,(y{\cdot}Q\widehat{\beta}) & \rightarrow & y{\cdot}(P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} & (x \neq y) \\ P\widehat{\alpha} \dagger \widehat{x}\,(x{\cdot}Q\widehat{\beta}) & \rightarrow & P\widehat{\alpha} \dagger \widehat{y}\,(y{\cdot}(P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta}) & (y\,\textit{fresh}, x\;\textit{not introduced}) \\ P\widehat{\alpha} \dagger \widehat{x}\,(\widehat{y}Q{\cdot}\beta) & \rightarrow & \widehat{y}(P\widehat{\alpha} \dagger \widehat{x}Q){\cdot}\beta & \end{array}$$

and the contextual rules

$$P \rightarrow Q \;\Rightarrow\; \begin{cases} y{\cdot}P\widehat{\alpha} \;\rightarrow\; y{\cdot}Q\widehat{\alpha} \\ \widehat{y}P{\cdot}\alpha \;\rightarrow\; \widehat{y}Q{\cdot}\alpha \end{cases}$$

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:

Theorem 7.5 *If $P\widehat{\alpha} \dagger \widehat{x}Q : \Gamma \vdash_{\textbf{LK}} \Delta$, then $P\widehat{\alpha} \dagger \widehat{x}Q$ can be contracted.*

Proof: Easy. □

We can show:

Theorem 7.6 (Witness reduction) *If $P : \Gamma \vdash_{\mathrm{LK}} \Delta$ and $P \to_{\mathcal{LK}} Q$, then $Q : \Gamma \vdash_{\mathrm{LK}} \Delta$.*

Proof: We just show the cases for some of the added rules; as mentioned above, the proof for the other rules can be found in [9].

$(\widehat{y}P\cdot\beta)\widehat{\beta} \dagger \widehat{x}(x\cdot Q\widehat{\alpha}) \to Q\widehat{\alpha} \dagger \widehat{y}P$: If $(\widehat{y}P\cdot\beta)\widehat{\beta} \dagger \widehat{x}(x\cdot Q\widehat{\alpha}) : \Gamma \vdash_{\mathrm{LK}} \Delta$, then the derivation is shaped like:

$$\dfrac{\dfrac{\begin{array}{c}\boxed{\phantom{P:\Gamma,y{:}A\vdash\Delta}}\\ P:\Gamma,y{:}A\vdash\Delta\end{array}}{\widehat{y}P\cdot\beta:\Gamma\vdash\beta{:}\neg A,\Delta}\,(r\text{-}inv)\qquad \dfrac{\begin{array}{c}\boxed{\phantom{Q:\Gamma\vdash\alpha{:}A,\Delta}}\\ Q:\Gamma\vdash\alpha{:}A,\Delta\end{array}}{x\cdot Q\widehat{\alpha}:\Gamma,x{:}\neg A\vdash\Delta}\,(l\text{-}inv)}{(\widehat{y}P\cdot\beta)\widehat{\beta}\dagger\widehat{x}(x\cdot Q\widehat{\alpha}):\Gamma\vdash\Delta}\,(cut)$$

Then we can construct:

$$\dfrac{\begin{array}{c}\boxed{\phantom{Q:\Gamma\vdash\alpha{:}A,\Delta}}\\ Q:\Gamma\vdash\alpha{:}A,\Delta\end{array}\qquad\begin{array}{c}\boxed{\phantom{P:\Gamma,y{:}A\vdash\Delta}}\\ P:\Gamma,y{:}A\vdash\Delta\end{array}}{Q\widehat{\beta}\dagger\widehat{x}P:\Gamma\vdash\Delta}\,(cut)$$

$(\widehat{y}Q\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}P \to (\widehat{y}(Q\widehat{\alpha}\dagger\widehat{x}P)\cdot\beta)\widehat{\beta}\dagger\widehat{x}P$ (β fresh, α *not introduced*): The derivation for the left-hand side is shaped like:

$$\dfrac{\dfrac{\begin{array}{c}\boxed{\phantom{Q:\Gamma,y{:}A\vdash\alpha{:}\neg A,\Delta}}\\ Q:\Gamma,y{:}A\vdash\alpha{:}\neg A,\Delta\end{array}}{\widehat{y}Q\cdot\alpha:\Gamma\vdash\alpha{:}\neg A,\Delta}\,(r\text{-}inv)\qquad\begin{array}{c}\boxed{\phantom{P:\Gamma,x{:}\neg A\vdash\Delta}}\\ P:\Gamma,x{:}\neg A\vdash\Delta\end{array}}{(\widehat{y}Q\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}P:\Gamma\vdash\Delta}\,(cut)$$

Then we can construct:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\boxed{\phantom{Q:\Gamma,y{:}A\vdash\alpha{:}\neg A,\Delta}}\\ Q:\Gamma,y{:}A\vdash\alpha{:}\neg A,\Delta\end{array}\qquad\dfrac{\begin{array}{c}\boxed{\phantom{P,x{:}\neg A:\Gamma\vdash\Delta}}\\ P,x{:}\neg A:\Gamma\vdash\Delta\end{array}}{P:\Gamma,x{:}\neg A,y{:}A\vdash\Delta}\,(Wk)}{Q\widehat{\alpha}\dagger\widehat{x}P:\Gamma,y{:}A\vdash\Delta}\,(cut)}{\widehat{y}(Q\widehat{\alpha}\dagger\widehat{x}P)\cdot\beta:\Gamma\vdash\beta{:}\neg A,\Delta}\,(r\text{-}inv)\qquad\begin{array}{c}\boxed{\phantom{P:\Gamma,x{:}\neg A\vdash\Delta}}\\ P:\Gamma,x{:}\neg A\vdash\Delta\end{array}}{(\widehat{y}(Q\widehat{\alpha}\dagger\widehat{x}P)\cdot\beta)\widehat{\beta}\dagger\widehat{x}P:\Gamma\vdash\Delta}\,(cut)$$

$P\widehat{\alpha}\dagger\widehat{x}(y\cdot Q\widehat{\beta}) \to y\cdot(P\widehat{\alpha}\dagger\widehat{x}Q)\widehat{\beta}$ $(x \neq y)$: The derivation for the left-hand side is shaped like:

$$\dfrac{\begin{array}{c}\boxed{\phantom{P:\Gamma,y{:}\neg B\vdash\alpha{:}A,\Delta}}\\ P:\Gamma,y{:}\neg B\vdash\alpha{:}A,\Delta\end{array}\qquad\dfrac{\begin{array}{c}\boxed{\phantom{Q:\Gamma,x{:}A\vdash\beta{:}B,\Delta}}\\ Q:\Gamma,x{:}A\vdash\beta{:}B,\Delta\end{array}}{y\cdot Q\widehat{\beta}:\Gamma,x{:}A,y{:}\neg B\vdash\Delta}\,(l\text{-}inv)}{P\widehat{\alpha}\dagger\widehat{x}(y\cdot Q\widehat{\beta}):\Gamma,y{:}\neg B\vdash\Delta}\,(cut)$$

and we can construct:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\boxed{\phantom{P:\Gamma,y{:}\neg B\vdash\alpha{:}A,\Delta}}\\ P:\Gamma,y{:}\neg B\vdash\alpha{:}A,\Delta\end{array}}{P:\Gamma,y{:}\neg B\vdash\alpha{:}A,\beta{:}B,\Delta}\,(Wk)\qquad\dfrac{\begin{array}{c}\boxed{\phantom{Q:\Gamma,x{:}A\vdash\beta{:}B,\Delta}}\\ Q:\Gamma,x{:}A\vdash\beta{:}B,\Delta\end{array}}{Q:\Gamma,y{:}\neg B,x{:}A\vdash\beta{:}B,\Delta}\,(Wk)}{P\widehat{\alpha}\dagger\widehat{x}Q:\Gamma,y{:}\neg B\vdash\beta{:}B,\Delta}\,(cut)}{y\cdot(P\widehat{\alpha}\dagger\widehat{x}Q)\widehat{\beta}:\Gamma,y{:}\neg B\vdash\Delta}\,(l\text{-}inv)$$

The other cases are similar. □

We will now extend the two translations so that we deal with the added connective as well.

Definition 7.7 (Negation) Negation gets represented in the π-calculus via the natural translation as:

$$\begin{array}{rcl} \llbracket x{\cdot}P\widehat{\alpha}\rrbracket_{\mathrm{N}} &=& x(\alpha).!\,\llbracket P\rrbracket_{\mathrm{N}} \\ \llbracket \widehat{x}P{\cdot}\alpha\rrbracket_{\mathrm{N}} &=& (\nu x)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid \overline{\alpha}\,x) \end{array}$$

via the semantic translation as:

$$\begin{array}{rcl} \llbracket x{\cdot}P\widehat{\alpha}\rrbracket_{\mathrm{S}} &=& !\,(\nu\alpha)\,(\llbracket P\rrbracket_{\mathrm{S}} \mid x(z).!\,\alpha{\rightarrow}\overline{z}) \\ \llbracket \widehat{x}P{\cdot}\alpha\rrbracket_{\mathrm{S}} &=& !\,(\nu x)\,(\llbracket P\rrbracket_{\mathrm{S}} \mid \overline{\alpha}\,x) \end{array}$$

This translation 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 natural and semantic translations of the witness for $\neg\neg A \rightarrow A$ now become:

$$\begin{array}{rcl} \llbracket \widehat{x}\,(x{\cdot}\,(\widehat{y}\langle y{\cdot}\alpha\rangle{\cdot}\gamma)\,\widehat{\gamma})\widehat{\alpha}{\cdot}\beta\rrbracket_{\mathrm{N}} &=& (\nu x\alpha)\,(!\,x(\gamma).!\,(\nu y)\,(!\,y(w).\overline{\alpha}\,w \mid \overline{\gamma}\,y) \mid \overline{\beta}\langle x,\alpha\rangle) \\ \llbracket \widehat{x}\,(x{\cdot}\,(\widehat{y}\langle y{\cdot}\alpha\rangle{\cdot}\gamma)\,\widehat{\gamma})\widehat{\alpha}{\cdot}\beta\rrbracket_{\mathrm{S}} &=& \\ & & !\,(\nu x\alpha)\,(!\,(\nu\gamma)\,(!\,(\nu y)\,(!\,y(w).\overline{\alpha}w \mid \overline{\gamma}\,y) \mid x(z).!\,\gamma{\rightarrow}\overline{z}) \mid \overline{\beta}\langle x,\alpha\rangle) \end{array}$$

The following consistency results are easy to prove.

Theorem 7.8 *Let* $(\widehat{y}P{\cdot}\beta)\,\widehat{\beta} \dagger \widehat{x}\,(x{\cdot}Q\widehat{\alpha})$ *be such that* β, x *are introduced. Then*

i) $\llbracket\,(\widehat{y}P{\cdot}\beta)\,\widehat{\beta} \dagger \widehat{x}\,(x{\cdot}Q\widehat{\alpha})\rrbracket_{\mathrm{N}} \;\underset{\sim}{\sqsupset}\; \llbracket Q\widehat{\alpha} \dagger \widehat{y}P\rrbracket_{\mathrm{N}}$.

ii) $\llbracket\,(\widehat{y}P{\cdot}\beta)\,\widehat{\beta} \dagger \widehat{x}\,(x{\cdot}Q\widehat{\alpha})\rrbracket_{\mathrm{S}} \;\approx\; \llbracket Q\widehat{\alpha} \dagger \widehat{y}P\rrbracket_{\mathrm{S}}$.

Proof: *i*)

$$\begin{array}{ll} \llbracket\,(\widehat{y}P{\cdot}\beta)\,\widehat{\beta} \dagger \widehat{x}\,(x{\cdot}Q\widehat{\alpha})\rrbracket_{\mathrm{N}} & \triangleq \\ (\nu x)\,(!\,(\nu y)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid \overline{x}\,y) \mid !\,x(\alpha).!\,\llbracket Q\rrbracket_{\mathrm{N}}) & \rightarrow_{\pi}\ (x) \\ (\nu y)\,(!\,\llbracket Q\rrbracket_{\mathrm{N}}[y/\alpha] \mid !\,\llbracket P\rrbracket_{\mathrm{N}}) \mid (\nu x)\,(!\,(\nu y)\,(!\,\llbracket P\rrbracket_{\mathrm{N}} \mid \overline{x}\,y) \mid !\,x(\alpha).!\,\llbracket Q\rrbracket_{\mathrm{N}}) & \triangleq \\ \llbracket Q\widehat{\alpha} \dagger \widehat{y}P\rrbracket_{\mathrm{N}} \mid \llbracket\,(\widehat{y}P{\cdot}\beta)\,\widehat{\beta} \dagger \widehat{x}\,(x{\cdot}Q\widehat{\alpha})\rrbracket_{\mathrm{N}} & \underset{\sim}{\sqsupset}\ \llbracket Q\widehat{\alpha} \dagger \widehat{y}P\rrbracket_{\mathrm{N}} \end{array}$$

ii)

$$\begin{array}{ll} \llbracket\,(\widehat{y}P{\cdot}\beta)\,\widehat{\beta} \dagger \widehat{x}\,(x{\cdot}Q\widehat{\alpha})\rrbracket_{\mathrm{S}} & \triangleq \\ !\,(\nu\beta x)\,(!\,(\nu y)\,(\llbracket P\rrbracket_{\mathrm{S}} \mid \overline{\beta}\,y) \mid !\,\beta{\rightarrow}\overline{x} \mid !\,(\nu\alpha)\,(\llbracket Q\rrbracket_{\mathrm{S}} \mid x(z).!\,\alpha{\rightarrow}\overline{z})) & \approx\ (\beta,x) \\ !\,(\nu\alpha y)\,(\llbracket Q\rrbracket_{\mathrm{S}} \mid !\,\alpha{\rightarrow}\overline{y} \mid \llbracket P\rrbracket_{\mathrm{S}}) & \triangleq\ \llbracket Q\widehat{\alpha} \dagger \widehat{y}P\rrbracket_{\mathrm{S}} \end{array}$$

□

We add the following type assignment rules for negation:

Definition 7.9 (Type assignment rules in $\vdash_{\pi}$ for $\neg$)

$$(\mathit{inv\text{-}r}) : \frac{}{\overline{a}\,x : x{:}A \vdash^{io}_{\pi} a{:}\neg A} \qquad\qquad (\mathit{inv\text{-}l}) : \frac{P : \Gamma \vdash^{io}_{\pi} x{:}A,\Delta}{a(x).P : \Gamma, a{:}\neg A \vdash^{io}_{\pi} \Delta}\ (a \notin \Gamma)$$

The correctness of the propagation rules follows as above in Theorems 4.8 and 5.4; notice that, since negation gets interpreted in the natural translation using input, the first contextual rule has to be excluded from $\rightarrow_{\mathrm{H}}$.

Theorem 7.10 $P : \Gamma \vdash_{\mathrm{LK}} \Delta$ *if and only if* $\llbracket P\rrbracket_{\mathrm{N}} : \Gamma \vdash^{io}_{\pi} \Delta$.

Proof: By induction on the structure of of terms in $\mathcal{LK}$; we only show the two added cases.

$x{\cdot}P\widehat{\alpha}$: Then $\llbracket x{\cdot}P\widehat{\alpha}\rrbracket_{\mathrm{N}} = x(\alpha).!\,\llbracket P\rrbracket_{\mathrm{N}}$, and, by induction, $P : \Gamma \vdash_{\mathrm{LK}} \Delta$ if and only if $\llbracket P\rrbracket_{\mathrm{N}} : \Gamma \vdash^{io}_{\pi} \Delta$. The last rule applied in the $\mathcal{LK}$-derivation is (*l-inv*):

$$\dfrac{\dfrac{\boxed{}}{P : \Gamma' \vdash \alpha{:}A,\Delta'}}{x{\cdot}P\widehat{\alpha} : \Gamma',x{:}\neg A \vdash \Delta'}\ (\textit{l-inv})$$

If $x(\alpha).!\,[\![P]\!]_{\rm N} : \Gamma \vdash_{\pi}^{io} \Delta$, then we know that x has a negated type,[18] and the derivation uses the rules (*inv-l*) and $(!)$, yielding

$$\dfrac{\dfrac{\dfrac{\boxed{}}{[\![P]\!]_{\rm N} : \Gamma \vdash_{\pi}^{io} \alpha{:}A,\Delta}}{!\,[\![P]\!]_{\rm N} : \Gamma \vdash_{\pi}^{io} \alpha{:}A,\Delta}\ (!)}{x(\alpha).!\,[\![P]\!]_{\rm N} : \Gamma,x{:}\neg A \vdash_{\pi}^{io} \Delta}\ (\textit{inv-l})$$

$\widehat{x}P{\cdot}\alpha$: Then $[\![\widehat{x}P{\cdot}\alpha]\!]_{\rm N} = (\nu x)\,(!\,[\![P]\!]_{\rm N} \,|\, \overline{\alpha}\,x)$, and, by induction, $P : \Gamma \vdash_{\rm LK} \Delta$ if and only if $[\![P]\!]_{\rm N} : \Gamma \vdash_{\pi}^{io} \Delta$ The last rule applied in the $\mathcal{LK}$-derivation is (*r-inv*):

$$\dfrac{\dfrac{\boxed{}}{P : \Gamma,x{:}A \vdash \Delta}}{\widehat{x}P{\cdot}\alpha : \Gamma \vdash \alpha{:}\neg A,\Delta}\ (\textit{r-inv})$$

If $(\nu x)\,(!\,[\![P]\!]_{\rm N} \,|\, \overline{\alpha}\,x) : \Gamma \vdash_{\pi}^{io} \Delta$, then we know that α has a negated type, and the derivation uses the rules ν, (*inv-l*), $(|)$, and $(!)$, yielding

$$\dfrac{\dfrac{\dfrac{\dfrac{\boxed{}}{[\![P]\!]_{\rm N} : \Gamma,x{:}A \vdash_{\pi}^{io} \Delta}}{!\,[\![P]\!]_{\rm N} : \Gamma,x{:}A \vdash_{\pi}^{io} \Delta}\ (!) \qquad \dfrac{}{\overline{\alpha}\,x : x{:}A \vdash_{\pi}^{io} \alpha{:}\neg A}\ (\textit{inv-r})}{!\,[\![P]\!]_{\rm N} \,|\, \overline{\alpha}\,x : \Gamma,x{:}A \vdash_{\pi}^{io} \alpha{:}\neg A,\Delta}\ (|)}{(\nu x)\,(!\,[\![P]\!]_{\rm N} \,|\, \overline{\alpha}\,x) : \Gamma \vdash_{\pi}^{io} \alpha{:}\neg A,\Delta}\ (\nu)$$

□

We can now check that the extended translation preserves assignable types as well; unlike in Theorem 6.8 and 6.9, the property is shown in only one direction. This is because, by not choosing new syntactic constructions to model negation, a process like $a(x).P$ not necessarily represents negation.

Theorem 7.11 *$P : \Gamma \vdash_{\rm LK} \Delta$ if and only if $[\![P]\!]_{\rm S} : \Gamma \vdash_{\pi}^{io} \Delta$.*

Proof: Much the same as that for Theorem 7.10. □

So our extended translations respect the classical sequent logic rules.

Conclusions

In this paper we have bridged the gap between classical *cut*-elimination and the semantics of concurrent calculi, by presenting translations of Gentzen's classical sequent calculus LK to the π-calculus that preserve *cut*-elimination. This was achieved through an embedding of the calculus $\mathcal{LK}$ into the π-calculus that translates a *cut* as synchronisation. $\mathcal{LK}$'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 $\mathcal{LK}$ can also not correspond to proofs.

$\mathcal{LK}$ introduces a natural concept of input and output that naturally translates into the input

[18] Notice that this would not be necessarily true for $x(\alpha).!\,P$, with P not the result of the interpretation.

and output primitives of the π-calculus. We presented two different translations, each with specific interesting properties. We first presented the natural translation, and showed that it preserves $\mathcal{LK}$'s head-reduction; in this translation we cannot represent full *cut*-elimination because we place some interpreted terms under input, in particular when interpreting the witness for $(\to L)$. This seems to be a natural consequence, and is a feature also in the various translations of the λ-calculus.

We then went on to show 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 translation and interpreting terms as infinite resources, we can represent full *cut*-elimination, but not through synchronisation, but rather weak bisimilarity. We have seen that this translation successfully represents Gentzen's *Hauptsatz* result, in that innermost reduction on typeable terms terminates.

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 $\mathcal{LK}$, but also can model proofs in LK in all detail in such a way that *cut*-elimination is preserved by weak bisimilarity. We also represented negation in $\mathcal{LK}$ by extending the syntax and reduction rules, and extended our translations to deal with the added construct. 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 the other logical connectives.

References

[1] Abadi, M., Cardelli, L., Curien, P.-L., Lévy, J.-J.: Explicit Substitutions. Journal of Functional Programming **1**(4), 375–416 (1991)

[2] Abadi, M., Gordon, A.: A Calculus for Cryptographic Protocols: The Spi Calculus. In: Proceedings of the Fourth ACM Conference on Computer and Communications Security, pp. 36–47. ACM Press (1997). DOI 10.1145/266420.266432

[3] Abramsky, S.: The lazy lambda calculus. In: Research topics in functional programming, pp. 65–116. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA (1990)

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

[5] Ariola, Z., Herbelin, H.: Minimal Classical Logic and Control Operators. In: J. Baeten, J. Lenstra, J. Parrow, G. Woeginger (eds.) Proceedings of Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003, *Lecture Notes in Computer Science*, vol. 2719, pp. 871–885. Springer Verlag (2003). DOI 10.1007/3-540-45061-0_68

[6] Audebaud, P., Bakel, S. van: Understanding $\mathcal{X}$ with $\lambda\mu$. Consistent interpretations of the implicative sequent calculus in natural deduction (2006). *Manuscript*

[7] Bakel, S. van, Cardelli, L., Vigliotti, M.: 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). DOI http://arxiv.org/abs/1109.4817

[8] Bakel, S. van, Lengrand, S., Lescanne, P.: The language $\mathcal{X}$: Circuits, Computations and Classical Logic. In: M. Coppo, E. Lodi, G. Pinna (eds.) Proceedings of Ninth Italian Conference on Theoretical Computer Science (ICTCS'05), Siena, Italy, *Lecture Notes in Computer Science*, vol. 3701, pp. 81–96. Springer Verlag (2005)

[9] Bakel, S. van, Lescanne, P.: Computation with Classical Sequents. Mathematical Structures in Computer Science **18**, 555–609 (2008). DOI 10.1017/S0960129508006762

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

[11] Bakel, S. van, Vigliotti, M.: A logical interpretation of the λ-calculus into the π-calculus, preserving spine reduction and types. In: M. Bravetti, G. Zavattaro (eds.) Proceedings of 20th International Conference on Concurrency Theory (CONCUR'09), Bologna, Italy, *Lecture Notes in Computer Science*, vol. 5710, pp. 84 – 98. Springer Verlag (2009). DOI 10.1007/978-3-642-04081-8_7

[12] Bakel, S. van, Vigliotti, M.: An Output-Based Semantics of $\lambda\mu$ with Explicit Substitution in the π-calculus - Extended Abstract. In: J.M. Baeten, T. Ball, F. de Boer (eds.) Theoretical Computer Science - 7th IFIP TC 1/WG 2.2 International Conference (TCS 2012), *Lecture Notes in Computer Science*, vol. 7604, pp. 372–387. Springer Verlag (2012). DOI 10.1007/978-3-642-33475-7_26

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

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

[15] Beffara, E.: Logique, réalisabilité et concurrence. Ph.D. thesis, Université Paris 7 (2005)

[16] Beffara, E.: Functions as proofs as processes. Technical report hal-00609866, Institut de Mathématiques de Luminy (2007)

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

[18] Bloo, R., Rose, K.: Preservation of Strong Normalisation in Named Lambda Calculi with Explicit Substitution and Garbage Collection. In: CSN'95 – Computer Science in the Netherlands, pp. 62–72 (1995). DOI 10.1.1.51.5026

[19] Bruijn, N. de: 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)

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

[21] Bruscoli, P., Guglielmi, A.: A Linear Logic Programming Language with Parallel and Sequential Conjunction. In: M. Alpuente, M. Sessa (eds.) Joint Conference on Declarative Programming (GULP-PRODE'95), Marina di Vietri, Italy, pp. 409–420 (1995)

[22] Caires, L., Pfenning, F.: Session Types as Intuitionistic Linear Propositions. In: P. Gastin, F. Laroussinie (eds.) Concurrency Theory, 21th International Conference, (CONCUR'10), Paris, France, 2010, *Lecture Notes in Computer Science*, vol. 6269, pp. 222–236. Springer Verlag (2010)

[23] Cimini, M., Coen, C. S., Sangiorgi, D.: Functions as Processes: Termination and the $\lambda\mu\tilde{\mu}$-Calculus. In: M. Wirsing, M. Hofmann, A. Rauschmayer (eds.) Trustworthly Global Computing - 5th International Symposium, TGC 2010, Munich, Germany, February 24-26, 2010, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 6084, pp. 73–86. Springer Verlag (2010). DOI 10.1007/978-3-642-15640-3_5

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

[25] Crolard, T.: A confluent lambda-calculus with a catch/throw mechanism. Journal of Functional Programming **9**(6), 625–647 (1999)

[26] Curien, P.-L., Herbelin, H.: The Duality of Computation. In: Proceedings of the 5th ACM SIGPLAN International Conference on Functional Programming (ICFP'00), *ACM Sigplan Notices*, vol. 35.9, pp. 233–243. ACM (2000). DOI 10.1145/351240.351262

[27] Curry, H., Feys, R.: Combinatory Logic, vol. 1. North-Holland, Amsterdam (1958)

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

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

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

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

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

[33] Herbelin, H.: 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 (1995)

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

[35] Honda, K., Laurent, O.: An exact correspondence between a typed pi-calculus and polarised proof-nets. Theoretical Computer Science **411**(22-24), 2223–2238 (2010)

[36] Honda, K., Tokoro, M.: An Object Calculus for Asynchronous Communication. In: P. America (ed.) ECOOP'91 European Conference on Object-Oriented Programming, Geneva, Switzerland, July 15-19, 1991, Proceedings, *Lecture Notes in Computer Science*, vol. 512, pp. 133–147. Springer Verlag (1991). DOI 10.1007/BFb0057019

[37] Honda, K., Yoshida, N.: On the reduction-based process semantics. Theoretical Computer Science **151**, 437–486 (1995)

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

[39] Kleene, S.: Introduction to Metamathematics. North Holland, Amsterdam (1952)

[40] Klop, J.: Term Rewriting Systems. In: S. Abramsky, D. Gabbay, T. Maibaum (eds.) Handbook of Logic in Computer Science, vol. 2, chap. 1, pp. 1–116. Clarendon Press (1992)

[41] Laurent, O.: Polarized proof-nets and $\lambda\mu$-calculus. Theoretical Computer Science **290**(1), 161–188 (2003)

[42] Milner, R.: Functions as Processes. Mathematical Structures in Computer Science **2**(2), 269–310 (1992). DOI 10.1017/S0960129500001407

[43] Parigot, M.: An algorithmic interpretation of classical natural deduction. In: Proceedings of 3rd International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'92), *Lecture Notes in Computer Science*, vol. 624, pp. 190–201. Springer Verlag (1992). DOI 10.1007/BFb0013061

[44] Sangiorgi, D., Walker, D.: The Pi-Calculus. Cambridge University Press (2001)

[45] Summers, A.: Curry-Howard Term Calculi for Gentzen-Style Classical Logic. Ph.D. thesis, Imperial College London (2008)

[46] Urban, C.: Classical Logic and Computation. Ph.D. thesis, University of Cambridge (2000)

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

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

[49] Wadler, P.: Call-by-Value is Dual to Call-by-Name. In: Proceedings of the eighth ACM SIGPLAN international conference on Functional programming, pp. 189 – 201 (2003)