On the Power of Name-Passing Communication[*]

Yuxi Fu

BASICS, Shanghai Jiao Tong University, China
fu-yx@cs.sjtu.edu.cn

Abstract

It is shown that generally higher order process calculi cannot be interpreted in name-passing calculi in a robust way.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.2 Modes of Computation

Keywords and phrases Interaction theory, expressiveness, bisimulation

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2017.22

1 Introduction

Many process calculi have been proposed [19]. It is important to investigate the relative expressiveness between them [29, 25, 18, 20, 23, 10, 8]. It is even more important to study the relative expressiveness between classes of these models [5]. Positive results can tell us which model plays a foundational role in interaction theory and which acts as a prototype in applications. A negative result often reveals a deep distinct property of a model we probably were not aware of beforehand.

There are basically three classes of interactively Turing powerful models characterized by the content of communication. In the name-passing calculi the messages are channels (channel names) [16, 28, 22]. The power of this class of process calculi crucially depends on the ability to pass around private channels. The general view is that the π-calculus is the "λ-calculus" of concurrency theory [15] and is capable of modelling all kinds of phenomena in both theory and application [34]. The second class contains higher order process calculi. The pure process-passing calculus is too weak [5]. The messages in higher order process calculi are normally abstractions [25, 26, 31, 32]. A receiving process can instantiate the parameters of the received abstraction with its own private channels. A seemingly exception is Thomsen's CHOCS [29, 30]. However there is a very strong operator in CHOCS, the so-called relabelling operator [14]. If we think of it the relabelling operator can achieve the effect of instantiations of abstractions. So in CHOCS process-passing is abstraction-passing in disguise. Concerning the relative expressive power of the higher order process calculi, it is a popular belief that they can be completely interpreted in the name-passing calculi. If the messages are neither channels nor abstractions, they can always be coded up by natural numbers. The class of value-passing calculi consists of those models whose contents of communications are numbers [13, 14, 11, 12]. It is proved in [5] that value-passing calculi and name-passing calculi are incompatible in expressive power. This result appears somewhat surprising at first sight. The value-passing calculi turns out to be very expressive. A great deal of higher order communications can be interpreted by value-passing communications [3]. More precisely, if in a higher order process calculus messages may contain neither unbound private channels nor global channels, then the calculus is a submodel of the standard value-passing calculus [3]. The intuition is

[*] The support from NSFC (61472239, ANR 61261130589) is gratefully acknowledged.

that such messages can be coded up by numbers. Therefore abstraction-passing becomes value-passing. Moreover the Gödel encoding is bijective. So full abstraction poses no problem. What we will show in this short paper, by generalizing the proof in [5], is that even such restricted higher order communications cannot be interpreted by the name-passing calculi in a faithful manner. This result contributes to a better understanding of the relative expressive power of the three classes of interaction models.

Section 2 introduces the criteria for relative expressiveness. Section 3 fixes notations for the π-calculus. Section 4 proves a general negative result about non-interpretability in the π-calculus. Section 5 derives the main result. Section 6 makes some concluding comments.

2 Relative Expressiveness

Given two models $\mathbb{M}, \mathbb{N}$, in what sense is $\mathbb{N}$ at least as expressive as $\mathbb{M}$? Different researchers have different criteria. In logic terms the answer is given by submodel relationship. In what follows we shall motivate a set of criteria that will uncover our definition of submodel relationship.

Generally we understand an interpretation of $\mathbb{M}$ in $\mathbb{N}$ as a binary relation $\propto$ from the set of $\mathbb{M}$-processes to the set of $\mathbb{N}$-processes. The interpretation should be *total* in the sense that for every $\mathbb{M}$-process M there is an $\mathbb{N}$-process N such that $M \propto N$. The interpretation should not introduce extra divergence. There should not be an infinite internal action sequence $N \xrightarrow{\tau} N_0 \xrightarrow{\tau} N_1 \xrightarrow{\tau} N_2 \xrightarrow{\tau} \ldots$ such that $M \propto N_i$ for all $i \in \omega$ if M cannot do any infinite internal action sequence. More formally this energy respect criterion is introduced in [24].

▶ **Definition 1.** The relation $\propto$ is *codivergent* if $M_0 \propto N_0$ implies the following: (i) If there is an infinite internal action sequence $N_0 \xrightarrow{\tau} N_1 \xrightarrow{\tau} N_2 \xrightarrow{\tau} \ldots$, there is some M' such that $M \xrightarrow{\tau} M' \propto N_i$ for some $i > 0$. (ii) If $M_0 \xrightarrow{\tau} M_1 \xrightarrow{\tau} M_2 \xrightarrow{\tau} \ldots$ is an infinite internal action sequence, there is some N' such that $N \xrightarrow{\tau} N'$ and $M_j \propto N'$ for some $j > 0$.

Codivergence is a computational property. The unique equality for computation, the extensional equality, is divergence sensitive. Codivergence is the divergence sensitive property reformulated in the presence of bisimulation. Bisimulation property [15, 21] is in essence also about computations [4]. This is particularly evident in the following definition, where $\Longrightarrow$ is the reflexive and transitive closure of the one step internal transition $\xrightarrow{\tau}$.

▶ **Definition 2.** The relation $\propto$ is a *bisimulation* if the following are valid whenever $M \propto N$:

1. If $M \xrightarrow{\tau} M'$, then
   a. either $N \Longrightarrow N'$ for some N' such that $M \propto N'$ and $M' \propto N'$, or
   b. $N \Longrightarrow N'' \xrightarrow{\tau} N'$ for some N'', N' such that $M \propto N''$ and $M' \propto N'$.
2. If $N \xrightarrow{\tau} N'$, then
   a. either $M \Longrightarrow M'$ for some M' such that $M' \propto N$ and $M' \propto N'$, or
   b. $M \Longrightarrow M'' \xrightarrow{\tau} M'$ for some M'', M' such that $M'' \propto N$ and $M' \propto N'$.

Case (1a) and case (2a) are about *deterministic* computation whereas case (1b) and case (2b) are to do with *nondeterministic* computation. The reader must have noticed that this is the branching bisimulation defined in [33]. The reason for using branching bisimulation is that even if we allow the interpretation N of M to carry out some internal adjustments (actions), we do not allow N to introduce any extra nondeterministic computation step. If $N \overset{\tau}{\Longrightarrow} N'$ for some N' not equivalent to N, then N' cannot be an interpretation of M. This would lead to a contradiction if M cannot do any nondeterministic computation. The problem with the weak bisimulation [14] is that it confuses nondeterministic computation with deterministic computation. This is a confusion that never happens in computation theory.

We also need criteria that take into account of interaction. We assume that all external actions admitted in all our models are carried out at global channels. We say that a process P is *observable* at a global channel a if $P \Longrightarrow P'$ for some P' such that P' may perform an external action at channel a. If N is an interpretation of M then they should have the same capacity to communicate at any particular global channel, hence the following [17].

▶ **Definition 3.** The relation $\propto$ is *equipollent* if $M \propto N$ implies that, for every global channel a, M is observable at a if and only if N is observable at a.

If $M \propto N$ then there should be no difference between M and N detectable by equivalent observers from the respective models. In other words $\propto$ should be closed under the concurrent composition operator. Without this closure property it does not make sense to talk about relative expressiveness for interaction model.

▶ **Definition 4.** The relation $\propto$ is *extensional* if $M \,|\, M' \propto N \,|\, N'$ whenever $M \propto N$ and $M' \propto N'$.

Intuitively $\mathbb{M}$ is a submodel of $\mathbb{N}$ if for each $\mathbb{M}$-process M there is an $\mathbb{N}$-process N that is equal to M as it were. Obviously the equality on $\mathbb{M}$-processes must be a special submodel relationship.

▶ **Definition 5.** The *absolute equality* $=_{\mathbb{M}}$ on $\mathbb{M}$ is the largest binary relation on $\mathbb{M}$-processes that renders true the following statements.
1. It is reflexive.
2. It is extensional, equipollent, codivergent, bisimilar.

We often omit the subscript in $=_{\mathbb{M}}$. It is important that expressiveness and equality are defined in completely the same fashion. Otherwise it would not be a *submodel* relationship, confer Lemma 7.

▶ **Definition 6.** A *subbisimilarity* from $\mathbb{M}$ to $\mathbb{N}$ is a binary relation $\propto$ from $\mathbb{M}$-processes to $\mathbb{N}$-processes such that the following statements are valid.
1. It is total and *semantical*, the latter means that $M' = M \propto N$ implies $M' \propto N$.
2. It is extensional, equipollent, codivergent, bisimilar.

The first condition of Definition 6 is a generalization of the reflexivity condition of Definition 5. Like reflexivity it is a preliminary requirement. The semantical condition asks for nothing more than that the relation $\propto$ should be *syntax independent* when seen as an interpretation from $\mathbb{M}$ to $\mathbb{N}$. We say that $\mathbb{M}$ is a *submodel* of $\mathbb{N}$, notation $\mathbb{M} \sqsubseteq \mathbb{N}$, if there is a subbisimilarity from $\mathbb{M}$ to $\mathbb{N}$. The terminology is enforced by the following full abstraction lemma.

▶ **Lemma 7.** *Suppose $\propto$ is a subbisimilarity from $\mathbb{M}$ to $\mathbb{N}$, and $M \propto N$ and $M' \propto N'$. Then $M = M'$ if and only if $N = N'$.*

Proof. $N =_{\mathbb{N}} N'$ implies $M =_{\mathbb{M}} M'$ since the composition $\propto; =_{\mathbb{N}}; \propto^{-1}$ is a subbisimilarity, where $\propto^{-1}$ is the reverse relation of $\propto$. Conversely if $M =_{\mathbb{M}} M'$ then $M' \propto N$ by the semantical condition. It follows from $M' \propto N'$ and $M' \propto N$ that $N =_{\mathbb{N}} N'$ because $\propto; =_{\mathbb{N}}$ is a subbisimilarity. ◀

In the rest of the paper we say that $P \xrightarrow{\tau} P'$ is a *deterministic* computation step, notation $P \to P'$, if $P' = P$, and that it is a *nondeterministic* computation step, notation $P \xrightarrow{\iota} P'$, if $P' \neq P$. We will write $\to^{+}$ ($\to^{*}$) for the (reflexive and) transitive closure of $\to$. A τ-*descendant* of T is a term T' such that $T \Longrightarrow T'$. Every model has an unobservable process $\mathbf{0}$ whose every computation is finite and an unobservable process Ω whose every computation is infinite. It is easy to see that both $\mathbf{0}$ and Ω can only do deterministic computation. For every subbisimilarity $\propto$ it holds that $Q = \Omega$ whenever $\Omega \propto Q$ or $Q \propto \Omega$. Similarly $Q = \mathbf{0}$ whenever $\mathbf{0} \propto Q$ or $Q \propto \mathbf{0}$.

For more motivations to the absolute equality and the subbismilarity, consult [5].

3 Name-Passing Calculus

We fix in this section the target model [16]. Unlike in the original presentation we introduce syntactical distinction between global channels, private channels and channel variables.

- Let $\mathcal{C}_g$ be the set of global channels. The elements of $\mathcal{C}_g$ are denoted by a, b, c, d, e, f.
- Let $\mathcal{C}_p$ be the set of private channels. The elements of $\mathcal{C}_p$ are denoted by l, m, n, o, p, q.
- Let $\mathcal{C}_v$ be the set of channel variables. The elements of $\mathcal{C}_v$ are denoted by u, v, w, x, y, z.

The set $\mathcal{C}_g \cup \mathcal{C}_p \cup \mathcal{C}_v$ will be ranged over by μ, ν, and the set $\mathcal{C}_g \cup \mathcal{C}_p$ by α, β. The π-terms are constructed from the following grammar.

$$S, T \quad := \quad \sum_{i\in I} \mu(x).T_i \mid \sum_{i\in I} \overline{\mu}\nu_i.T_i \mid S \,|\, T \mid (p)T \mid [\mu{=}\nu]T \mid [\mu{\neq}\nu]T \mid !\mu(x).T.$$

In the above definition I is a finite indexing set. We abbreviate $\sum_{i\in I} \mu(x).T_i$ and $\sum_{i\in I} \overline{\mu}\mu_i'.T_i$ to $\mathbf{0}$ if $I = \emptyset$. If the size of I is one, we get input term $\mu(x).T$ and output term $\overline{\mu}\mu'.T$. We abbreviate $(p)\mu p.T$ to $\mu(p).T$, $a(x).T$ to $a.T$ if x does not appear in T, and $\overline{a}(q).T$ to $\overline{a}.T$ if q does not appear in T. The labeled transition semantics is defined by the following rules.

$$\frac{}{\sum_{i\in I} \alpha(x).T_i \xrightarrow{\alpha\beta} T_i\{\beta/x\}} \qquad \frac{}{\sum_{i\in I} \overline{\alpha}\beta_i.T_i \xrightarrow{\overline{\alpha}\beta_i} T_i} \qquad \frac{T \xrightarrow{\lambda} T' \quad p\notin\lambda}{(p)T \xrightarrow{\lambda} (p)T'} \qquad \frac{T \xrightarrow{\overline{\alpha}p} T'}{(p)T \xrightarrow{\overline{\alpha}(p)} T'}$$

$$\frac{S \xrightarrow{\alpha\beta} S' \quad T \xrightarrow{\overline{\alpha}\beta} T'}{S \,|\, T \xrightarrow{\tau} S' \,|\, T'} \qquad \frac{S \xrightarrow{\alpha p} S' \quad T \xrightarrow{\overline{\alpha}(p)} T'}{S \,|\, T \xrightarrow{\tau} (p)(S' \,|\, T')} \qquad \frac{S \xrightarrow{\lambda} S'}{S \,|\, T \xrightarrow{\lambda} S' \,|\, T}$$

$$\frac{T \xrightarrow{\lambda} T'}{[\alpha{=}\alpha]T \xrightarrow{\lambda} T'} \qquad \frac{T \xrightarrow{\lambda} T'}{[\alpha{\neq}\beta]T \xrightarrow{\lambda} T'} \qquad \frac{}{!\alpha(x).T \xrightarrow{\alpha\beta} T\{\beta/x\} \,|\, !\alpha(x).T}$$

All symmetric rules are omitted. In the third rule $p{\notin}\lambda$ means that p does not appear in λ. In the transition $T \xrightarrow{\overline{\alpha}(p)} T'$ the action $\overline{\alpha}(p)$ is a bound output action. A π-process is a π-term in which all channel variables are bounded by input prefix operators and all private channels are bounded by scope operators. We write L, M, N, O, P, Q for processes. We write λ_a for an input or output action at channel a and $\overline{\lambda_a}$ for the complementary action. We will find it necessary to use the internal choice $\tau.S + \tau.T$ defined by $(p)(\overline{p} \,|\, p.S \,|\, p.T)$.

The above semantics is called early semantics in literature. There is also a late semantics in which instantiation comes after the commitment of an input action [16]. If we think of bisimulation as a computational property, only early semantics makes sense. However it must be said that, due to the ever presence of name extrusion in π, the late semantics is intrinsic to the π-calculus [27]. We explain the claim by an example. The numbers fetchable at channel α can be defined in π as follows.

$$\begin{aligned} [\![0]\!]_\alpha &\stackrel{\text{def}}{=} \overline{\alpha}(o).\overline{o}(p).\overline{o}(q).\overline{q}, \\ [\![k{+}1]\!]_\alpha &\stackrel{\text{def}}{=} \overline{\alpha}(o).\overline{o}(p).\overline{o}(q).\overline{p}(r).[\![k]\!]_r. \end{aligned}$$

It is simple to define the process $a(x).\mathit{if}\, x{=}0\ \mathit{then}\ P\ \mathit{else}\ Q$ that turns into P if the process has fetched a zero at channel a and turns into Q if the number it imports is not zero. Now consider $N \overset{\text{def}}{=} a(x).\big(\mathit{if}\, x{=}0\ \mathit{then}\, \overline{c}x\ \mathit{else}\, \overline{d}x\big)$, $C \overset{\text{def}}{=} !a(x).\overline{c}x$ and $D \overset{\text{def}}{=} !a(x).\overline{d}x$. It is easy to see that the equality

$$C \,|\, D \,|\, N \quad = \quad C \,|\, D \tag{1}$$

fails for the π-calculus. The observer O defined in (2) can make a choice between 0 and 1 after it is engaged.

$$O \overset{\text{def}}{=} a.b \,|\, \overline{a}(o).\, (\tau.\overline{o}(p).\overline{o}(q).\overline{q} + \tau.\overline{o}(p).\overline{o}(q).\overline{p}(r).[\![0]\!]_r)\,. \tag{2}$$

Let $C \,|\, D \,|\, N \,|\, O \xrightarrow{\tau} C \,|\, D \,|\, (o)(N' \,|\, O')$ be caused by the interaction between N and O. This internal action cannot be bisimulated vacuously by $C \,|\, D \,|\, O$ because $C \,|\, D \,|\, O$ is observable at b whereas $C \,|\, D \,|\, (o)(N' \,|\, O')$ is not. It cannot be bisimulated by $C \,|\, D \,|\, O \xrightarrow{\tau} (o)(C' \,|\, D \,|\, O'')$ caused by the interaction between C and O since $(o)(C' \,|\, D \,|\, O'')$ is not observable at d. Similarly the internal action of $C \,|\, D \,|\, O$ caused by the interaction between D and O does not bisimulate $C \,|\, D \,|\, N \,|\, O \xrightarrow{\tau} C \,|\, D \,|\, (o)(N' \,|\, O')$. It is worth pointing out that (1) fails also for weak bisimilarity.

A lot can happen between the time a π-process starts to fetch a message and the time it sees the full picture of the message. This is the fundamental weakness of name-passing communications. In other type of model there are more efficient implementations of $a(x).\mathit{if}\ x{=}0\ \mathit{then}\ P\ \mathit{else}\ Q$ that render (1) true. This simple example provides all the intuition for the main result of the paper.

The absolute equality for π-calculus can be equivalently defined in terms of input-output behaviours. Let's call a π-term a *quasi π-process* if it does not contain any free channel variables.

▶ **Definition 8.** A codivergent bisimulation $\mathcal{R}$ on quasi π-processes is an *extensional bisimulation* if the following statements are valid for all $\lambda \neq \tau$ whenever $S\mathcal{R}T$:

1. If $S \xrightarrow{\lambda} S'$, then $T \Longrightarrow T'' \xrightarrow{\lambda} T'$ for some T'', T' such that $S\mathcal{R}T''$ and $S'\mathcal{R}T'$.
2. If $T \xrightarrow{\lambda} T'$, then $S \Longrightarrow S'' \xrightarrow{\lambda} S'$ for some S'', S' such that $S''\mathcal{R}T$ and $S'\mathcal{R}T'$.

The *extensional bisimilarity* $\simeq_\pi$ is the largest extensional bisimulation.

Clause (1) and clause (2) of the above definition ensure that equal processes have the same input-output behaviours. The next lemma points out the authoritative role of the absolute equality. It is the minimal equality for interactive objects that subsumes the extensional equality of computation.

▶ **Lemma 9.** *Suppose P, Q are π-processes. Then $P =_\pi Q$ if and only if $P \simeq_\pi Q$.*

Proof. The relation $\simeq_\pi$ is closed under concurrent composition and scope operation. Hence $\simeq_\pi \subseteq =_\pi$. Conversely we show that the relation

$$\left\{ (S,T) \;\middle|\; \begin{array}{l} S \text{ and } T \text{ are quasi } \pi \text{ processes}, \\ (p_1,\ldots,p_n)(\overline{a_1}p_1 \,|\, \ldots \,|\, \overline{a_n}p_n \,|\, S) =_\pi (p_1,\ldots,p_n)(\overline{a_1}p_1 \,|\, \ldots \,|\, \overline{a_n}p_n \,|\, T), \\ p_1,\ldots,p_n \text{ are all the unbound private channels appearing in } S \,|\, T, \\ \text{none of the global channels } a_1,\ldots,a_n \text{ appears in } S \,|\, T. \end{array} \right\}.$$

is an extensional bisimulation. For details see [9]. ◀

4 A General Negative Result

We argue in this section that roughly speaking if a model can release an infinite number of pairwise distinct complete messages, then the model cannot be interpreted in the π-calculus.

Suppose $\mathfrak{D} = \{\lceil 0\rfloor_a, \lceil 1\rfloor_a, \lceil 2\rfloor_a, \ldots, \lceil i\rfloor_a, \ldots\}$ is an infinite set of processes in some model $\mathbb{M}$ such that $i \neq j$ implies

$$\lceil i\rfloor_a \neq \lceil j\rfloor_a. \tag{3}$$

We assume that for every i the process $\lceil i\rfloor_a$ can do one and only one action, which is an output action at channel a, and turns into a process equivalent to $\mathbf{0}$ after the action. In other words,

$$\lceil i\rfloor_a \xrightarrow{\overline{a}(\widehat{i})} = \mathbf{0}, \tag{4}$$

where $\widehat{i}$ is the message released by $\lceil i\rfloor_a$. There is no specific requirement on the messages $\widehat{0}, \widehat{1}, \widehat{2}, \ldots$ apart from that they are pairwise distinct. We require that in $\mathbb{M}$ we can define an absorbing process $\lceil \lambda x.\mathbf{0}\rfloor_a$, a successor process $\lceil \lambda x.x^+\rfloor_a$, a choice process $\lceil \lambda x.\tau.x^+{+}\tau\rfloor_a$, and a test process $\lceil \lambda x.x \overset{?}{>} k\rfloor_a$ for each $k \in \omega$. For each $i \in \omega$ each of the processes can carry out one and only one action. Their operational behaviours are specified as follows:

$$\begin{aligned}
\lceil \lambda x.\mathbf{0}\rfloor_a &\xrightarrow{a(\widehat{i})}= \mathbf{0}, && (5)\\
\lceil \lambda x.x^+\rfloor_a &\xrightarrow{a(\widehat{i})}= \lceil i{+}1\rfloor_a, && (6)\\
\lceil \lambda x.\tau.x^+{+}\tau\rfloor_a &\xrightarrow{a(\widehat{i})}= \tau.\lceil i{+}1\rfloor_a + \tau, && (7)\\
\lceil \lambda x.x \overset{?}{>} k\rfloor_a &\xrightarrow{a(\widehat{i})}= \mathbf{0}, \quad \text{if } i \leq k, && (8)\\
\lceil \lambda x.x \overset{?}{>} k\rfloor_a &\xrightarrow{a(\widehat{i})}= \lceil i{+}1\rfloor_a, \quad \text{if } i > k. && (9)
\end{aligned}$$

In $\mathbb{M}$ there is also a replicated form for each of $\lceil \lambda x.\mathbf{0}\rfloor_a$, $\lceil \lambda x.x^+\rfloor_a$ and $\lceil \lambda x.\tau.x^+{+}\tau\rfloor_a$, denoted respectively by $\lceil !\lambda x.\mathbf{0}\rfloor_a$, $\lceil !\lambda x.x^+\rfloor_a$ and $\lceil !\lambda x.\tau.x^+{+}\tau\rfloor_a$. For each $i \in \omega$ their unique actions are described as follows:

$$\begin{aligned}
\lceil !\lambda x.\mathbf{0}\rfloor_a &\xrightarrow{a(\widehat{i})}= \lceil !\lambda x.\mathbf{0}\rfloor_a, && (10)\\
\lceil !\lambda x.x^+\rfloor_a &\xrightarrow{a(\widehat{i})}= \lceil i{+}1\rfloor_a \,|\, \lceil !\lambda x.x^+\rfloor_a, && (11)\\
\lceil !\lambda x.\tau.x^+{+}\tau\rfloor_a &\xrightarrow{a(\widehat{i})}= (\tau.\lceil i{+}1\rfloor_a + \tau) \,|\, \lceil !\lambda x.\tau.x^+{+}\tau\rfloor_a. && (12)
\end{aligned}$$

Let G be the process $\lceil !\lambda x.\mathbf{0}\rfloor_a \,|\, \lceil !\lambda x.x^+\rfloor_a \,|\, \lceil !\lambda x.\tau.x^+{+}\tau\rfloor_a$. We assume that in $\mathbb{M}$ the following semantic equality is valid for each $k \in \omega$.

$$\Omega \,|\, G = \Omega \,|\, G \,|\, \lceil \lambda x.x \overset{?}{>} k\rfloor_a. \tag{13}$$

Our assumption on $\mathbb{M}$ is very liberal. It asks for no more than a numerical system that validates (13).

▶ **Theorem 10.** $\mathbb{M} \not\sqsubseteq \pi$.

Proof. Suppose there were a subbisimilarity $\mathfrak{I}$ from the $\mathbb{M}$-processes to the π-processes. In what follows we write $M\mathfrak{I}P \not\rightarrow$ for $M\mathfrak{I}P$ and $P \not\rightarrow$. Now fix $a \in \mathcal{C}_g$. We fix the notations for the interpretations of the $\mathbb{M}$-processes just described.

- For each $i \in \omega$ let N_i be the π-process such that $\lceil i\rfloor_a \,\mathfrak{I}\, N_i \not\rightarrow$.
- Let C_0 be the π-process such that $\lceil \lambda x.\mathbf{0}\rfloor_a \,\mathfrak{I}\, C_0 \not\rightarrow$.
- Let S_0 be the π-process such that $\lceil \lambda x.x^+\rfloor_a \,\mathfrak{I}\, S_0 \not\rightarrow$.

- Let E_0 be the π-process such that $\lceil\lambda x.\tau.x^{+}+\tau\rfloor_a \,\Im\, E_0 \nrightarrow$.
- Let C be the π-process such that $\lceil!\lambda x.\mathbf{0}\rfloor_a \,\Im\, C \nrightarrow$.
- Let S be the π-process such that $\lceil!\lambda x.x^{+}\rfloor_a \,\Im\, S \nrightarrow$.
- Let E be the π-process such that $\lceil!\lambda x.\tau.x^{+}+\tau\rfloor_a \,\Im\, E \nrightarrow$.
- For each $k \in \omega$ let J_k be the π-process such that $\lceil\lambda x.x \overset{?}{>} k\rfloor_a \,\Im\, J_k \nrightarrow$.

It follows from (13) and extensionality that

$$\Omega \,|\, C \,|\, S \,|\, E = \Omega \,|\, C \,|\, S \,|\, E \,|\, J_k. \tag{14}$$

We now derive some properties for the π-processes $N_0, N_1, N_2, N_3, \ldots$.

1. According to codivergence property there is no infinite computation sequence from $S_0 \,|\, N_i$. Suppose $S_0 \,|\, N_i \to B$. This action must be caused by $S_0 \xrightarrow{\lambda_a} S'$ and $N_i \xrightarrow{\overline{\lambda_a}} N'$ for some complementary actions $\lambda_a, \overline{\lambda_a}$ and some π-processes S' and N'. If neither λ_a nor $\overline{\lambda_a}$ is a bound output action then $S_0 \,|\, N_i \xrightarrow{\lambda_a}\xrightarrow{\overline{\lambda_a}} B$ and $S_0 \,|\, N_i \xrightarrow{\overline{\lambda_a}}\xrightarrow{\lambda_a} B$. It follows that $B \to^{*}\xrightarrow{\lambda_a}\to^{*}\xrightarrow{\overline{\lambda_a}} B' = B$ and $B \to^{*}\xrightarrow{\overline{\lambda_a}}\to^{*}\xrightarrow{\lambda_a} B'' = B$ for some B', B''. Now $B \,|\, B \to^{+} B' \,|\, B'' = B \,|\, B$. We derive by induction that there would be an infinite computation sequence from $S_0 \,|\, N_i \,|\, S_0 \,|\, N_i$, contradicting to the fact that $\lceil\lambda x.x^{+}\rfloor_a \,|\, \lceil i\rfloor_a \,|\, \lceil\lambda x.x^{+}\rfloor_a \,|\, \lceil i\rfloor_a$ is not divergent. If one of $\lambda_a, \overline{\lambda_a}$ is a bound output action, we also get an infinite sequence of computation by renaming and inserting scope operators in appropriate places. More specifically without loss of generality suppose $\lambda_a = ao$ and $\overline{\lambda_a} = \overline{a}(o)$. Let $S_0 \xrightarrow{aq} B_1$ and $N_i \xrightarrow{\overline{a}(p)} B_2$ for fresh private channels p, q. Then
$$B \equiv (o)(B_1\{o/p\} \,|\, B_2\{o/q\}). \tag{15}$$
It follows from $S_0 \,|\, N_i = B$ that there must exist some B' such that
$$S_0 \,|\, N_i \xrightarrow{aq}\xrightarrow{\overline{a}(p)} B_1 \,|\, B_2. \tag{16}$$
is bisimulated by
$$B \to^{*}\xrightarrow{aq}\to^{*}\xrightarrow{\overline{a}(p)} B'. \tag{17}$$
Now (15) and (17) imply that (16) can be extended to
$$S_0 \,|\, N_i \xrightarrow{aq}\xrightarrow{\overline{a}(p)} B_1 \,|\, B_2 \Longrightarrow\xrightarrow{\lambda_a^1}\Longrightarrow\xrightarrow{\overline{\lambda_a^1}} B_1' \,|\, B_2'. \tag{18}$$
for some B_1', B_2' and some $\lambda_a^1, \overline{\lambda_a^1}$. Consequently the bisimulation (17) can be extended to
$$B \to^{*}\xrightarrow{aq}\to^{*}\xrightarrow{\overline{a}(p)} B' \Longrightarrow\xrightarrow{\lambda_a^1}\Longrightarrow\xrightarrow{\overline{\lambda_a^1}} B'' \tag{19}$$
for some B''. The extension can be repeated infinitely often. We eventually get an infinite sequence with alternating input-output actions. Similarly we can derive from $S_0 \,|\, N_i \xrightarrow{\overline{a}(p)}\xrightarrow{aq} B_1 \,|\, B_2$ an infinite sequence with alternating output-input actions. In this way $S_0 \,|\, N_i \,|\, S_0 \,|\, N_i$ would induce an infinite sequence of computation. This is again a contradiction. We conclude that $S_0 \,|\, N_i \xrightarrow{\iota}= N_{i+1}$ is essentially the only one-step nondeterministic computation of $S_0 \,|\, N_i$.
2. N_i cannot do both an input action at channel a and an output action at channel a. Otherwise $N_i \,|\, N_i$ would be able to do an interaction, which would be a contradiction. This is because $N_i \,|\, N_i$ cannot perform any nondeterministic computation since $\lceil i\rfloor_a \,|\, \lceil i\rfloor_a$ cannot do any internal action. It cannot do a deterministic computation step since that would induce an infinite sequence of internal actions from $N_i \,|\, N_i \,|\, N_i \,|\, N_i$, like in the previous case. So N_i may perform either an input action or an output action exclusively.

3. If N_i can do an input, respectively output action then N_j can do an input, respectively output action for all $j \in \omega$ since the latter has to interact with C_0.

4. It follows from $\lceil \lambda x.\mathbf{0} \rfloor_a \,|\, \lceil i \rfloor_a \xrightarrow{\iota}= \mathbf{0}$ that $C_0 \,|\, N_i \xrightarrow{\iota}= \mathbf{0}$. Suppose $N_i \xrightarrow{\overline{a}c} N'$ and $N_j \xrightarrow{\overline{a}c} N''$ for some N', N''. Clearly $N' = \mathbf{0} = N''$. But then one could derive from $N_i \,|\, S_0 \xrightarrow{\iota}= N_{i+1}$ and $N_j \,|\, S_0 \xrightarrow{\iota}= N_{j+1}$ that $N_{i+1} = N_{j+1}$, which implies $i = j$. We conclude that if both N_i and N_j can do free output actions at channel a then the channels they release must be distinct whenever $i \neq j$.

5. According to our assumption we have $S_0 \,|\, N_i \xrightarrow{\tau}= N_{i+1}$ for all $i \geq 0$. Let's write $P \xrightarrow{\lambda}$ if $P \xrightarrow{\lambda} P'$ for some P'. It should be clear that if $N_1 \xrightarrow{\overline{a}c}$, then $S_0 \,|\, N_0 \Longrightarrow\xrightarrow{\overline{a}c}$. Similarly if $N_2 \xrightarrow{\overline{a}d}$, then $S_0 \,|\, N_1 \Longrightarrow\xrightarrow{\overline{a}d}$. Therefore $S_0 \,|\, S_0 \,|\, N_0 \Longrightarrow\xrightarrow{\overline{a}d}$. By induction we can prove that if N_i can release a global channel at a then that global channel must appear in $S_0 \,|\, N_0$. So only a finite number of $N_0, N_1, N_2, \ldots$ can do free output actions. Let h be the least number such that, for every $j \geq h$, N_j does only a bound output action.

We prove the impossibility result by a case analysis on the actions of $N_0, N_1, N_2, N_3, \ldots$.

1. Suppose $k > h$. Let

$$\begin{array}{ccl} N_h & \xrightarrow{\overline{a}(p)} & N'_h, \\ N_k & \xrightarrow{\overline{a}(p)} & N'_k. \end{array}$$

In this case C_0, D_0, E_0, J_h, C, D and E can only do input actions at channel a. Let

$$\begin{array}{ccl} C_0 & \xrightarrow{ap} & C_0^p, \\ D_0 & \xrightarrow{ap} & S_0^p, \\ E_0 & \xrightarrow{ap} & E_0^p, \\ J_h & \xrightarrow{ap} & J_p, \\ C & \xrightarrow{ap} & C_p = C, \\ D & \xrightarrow{ap} & S_p = S_0^p \,|\, S, \\ E & \xrightarrow{ap} & E_p = E_0^p \,|\, E. \end{array}$$

None of C_0, D_0, E_0, J_h, C, D and E may perform any output actions. Otherwise there would be either an infinite deterministic computation or a nondeterministic computation step. It follows from (14) that $\Omega \,|\, C \,|\, S \,|\, E \,|\, J_h \xrightarrow{ap} \Omega \,|\, C \,|\, S \,|\, E \,|\, J_p$ must be bisimulated by one of the following.

$$\begin{array}{ccl} \Omega \,|\, C \,|\, S \,|\, E & \xrightarrow{ap} & \Omega \,|\, C_p \,|\, S \,|\, E, \\ \Omega \,|\, C \,|\, S \,|\, E & \xrightarrow{ap} & \Omega \,|\, C \,|\, S_p \,|\, E, \\ \Omega \,|\, C \,|\, S \,|\, E & \xrightarrow{ap} & \Omega \,|\, C \,|\, S \,|\, E_p. \end{array}$$

In the first case $\Omega \,|\, C \,|\, S \,|\, E \,|\, J_h \,|\, N_k \xrightarrow{\iota} (p)(\Omega \,|\, C \,|\, S \,|\, E \,|\, J_p \,|\, N'_k)$ should be bisimulated by $\Omega \,|\, C \,|\, S \,|\, E \,|\, N_k \xrightarrow{\iota} (p)(\Omega \,|\, C_p \,|\, S \,|\, E \,|\, N'_k)$ due to congruence. This is impossible because

$$\begin{array}{rcl} (p)(\Omega \,|\, C \,|\, S \,|\, E \,|\, J_p \,|\, N'_k) & = & \Omega \,|\, C \,|\, S \,|\, E \,|\, N_{k+1} \\ & \neq & \Omega \,|\, C \,|\, S \,|\, E \\ & = & (p)(\Omega \,|\, C \,|\, S \,|\, E \,|\, N'_k) \\ & = & (p)(\Omega \,|\, C_p \,|\, S \,|\, E \,|\, N'_k) \end{array}$$

according to Lemma 7. In the second case $\Omega \,|\, C \,|\, S \,|\, E \,|\, J_h \,|\, N_h \xrightarrow{\iota} (p)(\Omega \,|\, C \,|\, S \,|\, E \,|\, J_p \,|\, N'_h)$ should be bisimulated by $\Omega \,|\, C \,|\, S \,|\, E \,|\, N_h \xrightarrow{\iota} (p)(\Omega \,|\, C \,|\, S_p \,|\, E \,|\, N'_h)$. Using again the full abstraction lemma one derives the following contradiction.

$$\begin{array}{rcl}(p)(\Omega \,|\, C \,|\, S \,|\, E \,|\, J_p \,|\, N'_h) & = & \Omega \,|\, C \,|\, S \,|\, E \\ & \neq & \Omega \,|\, C \,|\, S \,|\, E \,|\, N_{h+1} \\ & = & (p)(\Omega \,|\, C \,|\, S_0^p \,|\, S \,|\, E \,|\, N'_h) \\ & = & (p)(\Omega \,|\, C \,|\, S_p \,|\, E \,|\, N'_h).\end{array}$$

In the third case $\Omega \,|\, C \,|\, S \,|\, E \,|\, J_h \,|\, N_k \xrightarrow{\iota} (p)(\Omega \,|\, C \,|\, S \,|\, E \,|\, J_p \,|\, N'_k)$ should be bisimulated by $\Omega \,|\, C \,|\, S \,|\, E \,|\, N_k \xrightarrow{\iota} (p)(\Omega \,|\, C \,|\, S \,|\, E_p \,|\, N'_k)$. This is also impossible because $(p)(\Omega \,|\, C \,|\, S \,|\, E_p \,|\, N'_k) = (p)(\Omega \,|\, C \,|\, S \,|\, E_0^p \,|\, E \,|\, N'_k)$ can do a nondeterministic computation step, reaching to a state where N_{k+1} is *not* a concurrent component, while on the other hand $(p)(\Omega \,|\, C \,|\, S \,|\, E \,|\, J_p \,|\, N'_k) = \Omega \,|\, C \,|\, S \,|\, E \,|\, N_{k+1}$.

2. Now suppose the immediate actions of $N_0, N_1, N_2, \ldots$ are input actions. In this case an output action of $\Omega \,|\, C \,|\, S \,|\, E \,|\, J_h$ induced by J_h, whether it is a free output action or a bound output action, must be bisimulated by an output action of $\Omega \,|\, C \,|\, S \,|\, E$ induced by C or D or E. We can deduce a contradiction as in the first case.

We have proved that there cannot be any subbisimilarity from $\mathbb{M}$ to the π-calculus. ◀

5 Application to Higher Order Process Calculi

We demonstrate in this section that higher order process calculi cannot be interpreted in the π-calculus. Since the π-calculus is complete, it makes sense to focus on complete higher order process calculi. The completeness brings out the simplicity of the counter example and reveals the strength of the negative result.

Intuitively a model is complete if it is Turing powerful in an interactive fashion. A precise definition of completeness is given in terms of the *computability model* denoted by $\mathbb{C}$. The $\mathbb{C}$-processes are generated from the following grammar.

$$P \quad := \quad \mathbf{0} \mid \Omega \mid F_a^b(\mathsf{f}(x)) \mid \overline{a}(i) \mid P \,|\, P,$$

where f is a computable function and $i \in \omega$. The semantics is defined by the following rules, in which $\mathsf{f}(i)\!\uparrow$ means that f is undefined on i.

$$\frac{}{\Omega \xrightarrow{\tau} \Omega} \qquad \frac{}{\overline{a}(i) \xrightarrow{\overline{a}(i)} \mathbf{0}} \qquad \frac{}{F_a^b(\mathsf{f}(x)) \xrightarrow{a(i)} \overline{b}(j)}\ \mathsf{f}(i) = j \qquad \frac{}{F_a^b(\mathsf{f}(x)) \xrightarrow{a(i)} \Omega}\ \mathsf{f}(i)\!\uparrow$$

$$\frac{P \xrightarrow{\lambda} P'}{P \,|\, Q \xrightarrow{\lambda} P' \,|\, Q} \qquad \frac{P \xrightarrow{a(i)} P' \quad Q \xrightarrow{\overline{a}(i)} Q'}{P \,|\, Q \xrightarrow{\tau} P' \,|\, Q'}$$

The process Ω diverges. The process $F_a^b(\mathsf{f}(x))$ simulates the input-output behaviour of $\mathsf{f}(x)$ with input channel a and output channel b. The output process $\overline{a}(i)$ releases the number i at channel a. The model $\mathbb{C}$ is the minimal interactive extension of the computable functions. It says nothing about how computations are done. For this reason it is the best model to formalize the notion of completeness for interaction models. See [5] for more discussions and technical backgrounds.

▶ **Definition 11.** An interaction model $\mathbb{M}$ is *complete* if $\mathbb{C} \sqsubseteq \mathbb{M}$.

The completeness of the π-calculus and the completeness of value-passing calculi are proved in [5]. In the same paper it is shown that there is no subbisimilarity from a value-passing calculus to π. The reader should convince herself/himself that the value-passing processes satisfying (5) through (13) are easy to define.

We now convince the reader that the π-calculus cannot in general interpret higher order process calculi in a robust way. As an example we take a look at a particular functional model, denoted by $\mathbb{L}$. This is an abstraction-passing calculus. To define the model we need the syntactical class $\mathcal{V}$ of abstraction variables. The elements of $\mathcal{V}$ are denoted by U, V, W, X, Y, Z. The set of $\mathbb{L}$-*terms* is generated by the following grammar:

$$\begin{array}{rcl} S, T & := & \mathbf{0} \mid \mu(X).T \mid \overline{\mu}[A].T \mid S \,|\, T \mid (p)T \mid A(\mu,\nu), \\ A & := & X \mid \lambda(u,v).T. \end{array}$$

An abstraction is either an abstraction variable or of the form $\lambda(u,v).T$. The term $A(\alpha,\beta)$ is an instantiation of A at α,β. We have the grammar equality $(\lambda(u,v).T)(\alpha,\beta) \equiv A\{\alpha/u,\beta/v\}$. We abbreviate $\lambda(u,v).T$ to $\lambda(u).T$ if v does not appear in T and accordingly $A(\alpha,\alpha)$ to $A(\alpha)$. Because we would like to think that the abstraction A in $\overline{\mu}[A].T$ represents a complete message, we impose the following constraint.

(‡) In $\overline{\mu}[A].T$ the abstraction A contains no free channel variables, no unbound private channels and no occurrences of global channel.

The constraint (‡) gives rise to a simpler and weaker model whose algebraic property is easier to work out.

Using the action set $\{\alpha(A), \overline{\alpha}(A) \mid \alpha \in \mathcal{C}_g \cup \mathcal{C}_p\} \cup \{\tau\}$, the simple operational semantics of $\mathbb{L}$ is defined by the following rules. Notice that because of the constraint (‡) on abstractions there is no name extrusion.

$$\frac{}{\alpha(X).T \xrightarrow{\alpha(A)} T\{A/X\}} \qquad \frac{}{\overline{\alpha}[A].T \xrightarrow{\overline{\alpha}(A)} T} \qquad \frac{S \xrightarrow{\lambda} S'}{S \,|\, T \xrightarrow{\lambda} S' \,|\, T} \qquad \frac{S \xrightarrow{\alpha(A)} S' \quad T \xrightarrow{\overline{\alpha}(A)} T'}{S \,|\, T \xrightarrow{\tau} S' \,|\, T'}$$

$$\frac{S \xrightarrow{\lambda} T}{(p)S \xrightarrow{\lambda} (p)T}\ p \notin \lambda$$

The forever diverging process can be defined as follows.

$$\Omega \stackrel{\text{def}}{=} (p)\,(p(X).(Xp \,|\, \overline{p}[X]) \,|\, \overline{p}[\lambda(x).x(X).(Xx \,|\, \overline{x}[X])])\,. \tag{20}$$

To demonstrate the power of $\mathbb{L}$ we provide a direct interpretation of the lazy λ-calculus [1] in $\mathbb{L}$. Given an injective function from the set of λ variables to $\mathcal{V}$, a λ-term M is interpreted as an abstraction by the following structural induction.

$$\begin{array}{rcl} [\![x]\!] & = & X, \\ [\![\lambda x.M]\!] & = & \lambda(u,v).u(X).\overline{v}[[\![M]\!]], \\ [\![MN]\!] & = & \lambda(u,v).(mq)([\![M]\!](m,q) \,|\, \overline{m}[[\![N]\!]].q(Z).Z(u,v)). \end{array}$$

The process $[\![M]\!](a,b)$ is a "function" that inputs a "λ-term" at channel a and output the result "λ-term" at channel b. The structural aspect of the interpretation is enforced by the following simple lemma.

▶ **Lemma 12.** $[\![M\{N/X\}]\!] \equiv [\![M]\!]\{[\![N]\!]/X\}$.

Let $=_\beta$ denote the β-conversion. The interpretation is semantically correct, guaranteed by the following facts. In the statement of Lemma 13 we identify $\mathbf{0} \,|\, P$ to P syntactically.

▶ **Lemma 13.** *For closed λ-terms M, N, $M \rightarrow N$ if and only if $[\![M]\!](a,b) \rightarrow\rightarrow [\![N]\!](a,b)$.*

Proof. The λ-term M must be of the form $(\lambda x.L)M_1M_2\ldots M_k$, where the application is associative to the left. The encoding $[\![M]\!](a,b)$ is by definition of the form

$$..(m_2q_2)\,((m_1q_1)([\![\lambda x.L]\!](m_1,q_1)\,|\,\overline{m_1}[[\![M_1]\!]].q_1(Z).Z(m_2,q_2))\,|\,|\,\overline{m_2}[[\![M_2]\!]].q_2(Z).Z(m_3,q_3))\,..$$

which reduces in two deterministic computation steps to $[\![L\{M_1/x\}M_2\ldots M_k]\!](a,b) \equiv [\![N]\!](a,b)$ using Lemma 12. The argument is reversible. ◀

▶ **Lemma 14.** *For closed λ-terms M,N, $M =_\beta N$ implies $[\![M]\!](a,b) =_{\mathbb{L}} [\![N]\!](a,b)$.*

Proof. In view of Lemma 13 this is the Church-Rosser property [2]. ◀

It is well-known [2] that there is a λ-term encoding $\lceil 0\rceil, \lceil 1\rceil, \lceil 2\rceil, \ldots, \lceil i\rceil, \ldots$ of the natural numbers and an encoding $\lceil\_\rceil$ of the recursive functions such that $\lceil \mathsf{f}\rceil\lceil i\rceil \rightarrow^+ \lceil \mathsf{f}(i)\rceil$ if $\mathsf{f}(i)$ is defined, and that $\lceil \mathsf{f}\rceil\lceil i\rceil \rightarrow\rightarrow \ldots$ diverges if $\mathsf{f}(i)$ is undefined. Given global channels a,b we define the $\mathbb{L}$-processes

$$\overline{a}(i) \stackrel{\text{def}}{=} \overline{a}[[\![\lceil i\rceil]\!]], \tag{21}$$
$$\lambda_{a,b}(\mathsf{f}) \stackrel{\text{def}}{=} [\![\lceil \mathsf{f}\rceil]\!](a,b). \tag{22}$$

It is straightforward to verify that if $\mathsf{f}(i) = j$ then

$$\lambda_{a,b}(\mathsf{f})\,|\,\overline{a}(i) \xrightarrow{\iota}= \overline{b}(j) \tag{23}$$

and if $\mathsf{f}(i)$ is undefined then

$$\lambda_{a,b}(\mathsf{f})\,|\,\overline{a}(i) \xrightarrow{\iota}= \Omega. \tag{24}$$

The processes defined in (20), (21) and (22) actually provide an encoding of $\mathbb{C}$ into $\mathbb{L}$, hence the following.

▶ **Lemma 15.** $\mathbb{C} \sqsubseteq \mathbb{L}$.

Because of the constraint (‡) the completeness proof given in [32] cannot be carried out in $\mathbb{L}$. The replication operator defined in [32] does not seem to be encodable in $\mathbb{L}$.

To apply the general negative result it suffices to explain how numbers are defined in $\mathbb{L}$. Given an abstraction A we write A^+ for $\lambda(x,y).\overline{x}[A]$. Let

$$\widehat{0} \stackrel{\text{def}}{=} \lambda(x,y).\overline{y}[\mathbf{0}],$$
$$\widehat{k+1} \stackrel{\text{def}}{=} (\widehat{k})^+.$$

The abstractions $\widehat{0}, \widehat{1}, \widehat{2}, \ldots$ represent numbers. Using these numbers we can define the following simple processes.

$$
\begin{aligned}
\lceil i \rfloor_a &\overset{\text{def}}{=} \overline{a}[\widehat{i}],\\
\lceil \lambda x.\mathbf{0} \rfloor_a &\overset{\text{def}}{=} a(X),\\
\lceil \lambda x.x^+ \rfloor_a &\overset{\text{def}}{=} a(X).\overline{a}[X^+],\\
\lceil \lambda x.\tau.x^+{+}\tau \rfloor_a &\overset{\text{def}}{=} a(X).\left(\tau.\overline{a}[X^+] + \tau\right),\\
\lceil \lambda x.x \overset{?}{>} k \rfloor_a &\overset{\text{def}}{=} a(X).(n_1 o_1)(X(n_1,o_1) \,|\, n_1(X_1).(n_2 o_2)(X_1(n_2,o_2) \,|\, \ldots\\
&\qquad\quad |\, n_{k-1}(X_{k-1}).(n_k o_k)(X_{k-1}(n_k,o_k) \,|\, n_k.\overline{a}[X^+])\ldots)),\\
\lceil !\lambda x.\mathbf{0} \rfloor_a &\overset{\text{def}}{=} (p)\left(a(X).p(Z).(Z(a,p) \,|\, \overline{p}[Z]) \,|\, \overline{p}[\lambda(x,z).x(X).z(Z).(Z(x,z) \,|\, \overline{z}[Z])]\right),\\
\lceil !\lambda x.x^+ \rfloor_a &\overset{\text{def}}{=} (p)(a(X).(\overline{a}[X^+] \,|\, p(Z).(Z(a,p) \,|\, \overline{p}[Z]))\\
&\qquad\quad |\, \overline{p}[\lambda(x,z).x(X).(\overline{x}[X^+] \,|\, z(Z).(Z(x,z) \,|\, \overline{z}[Z])]))),\\
\lceil !\lambda x.\tau.x^+{+}\tau \rfloor_a &\overset{\text{def}}{=} (p)(a(X).((\tau.\overline{a}[X^+] + \tau) \,|\, p(Z).(Z(a,p) \,|\, \overline{p}[Z]))\\
&\qquad\quad |\, \overline{p}[\lambda(x,z).x(X).((\tau.\overline{x}[X^+] + \tau) \,|\, z(Z).(Z(x,z) \,|\, \overline{z}[Z])])).
\end{aligned}
$$

The last three processes, $\lceil !\lambda x.\mathbf{0} \rfloor_a$, $\lceil !\lambda x.x^+ \rfloor_a$ and $\lceil !\lambda x.\tau.x^+{+}\tau \rfloor_a$, are intuitively the processes $!a(X)$, $!a(X).\overline{a}[X^+]$ and $!a(X).\left(\tau.\overline{a}[X^+] + \tau\right)$. These replication processes must be implemented in $\mathbb{L}$. It is easy to verify that the processes defined in the above satisfy the necessary requirements stated in (3) through (12).

Let $G \overset{\text{def}}{=} \lceil !\lambda x.\mathbf{0} \rfloor_a \,|\, \lceil !\lambda x.x^+ \rfloor_a \,|\, \lceil !\lambda x.\tau.x^+{+}\tau \rfloor_a$. We now prove (13).

▶ **Lemma 16.** $\Omega \,|\, G = \Omega \,|\, G \,|\, \lceil \lambda x.x \overset{?}{>} k \rfloor_a$.

Proof. Consider the following transition

$$\Omega \,|\, G \,|\, \lceil \lambda x.x \overset{?}{>} k \rfloor_a \xrightarrow{a(A)} \Omega \,|\, G \,|\, J_A, \tag{25}$$

where J_A is the following process

$$(n_1 o_1)(A(n_1,o_1) \,|\, n_1(X_1).(n_2 o_2)(X_1(n_2,o_2) \,|\, \ldots \,|\, (X_{k-1}(n_k,o_k) \,|\, n_k.\overline{a}[A^+])\ldots)).$$

The process J_A and all its descendants can either carry out an internal action or enable $\overline{a}[A^+]$. They cannot do any other actions due to the constraint (‡). Let $\mathfrak{J}$ be the set of all terms J' such that $J_A \Longrightarrow J'$. It can be partitioned into three disjoint subsets.

$$
\begin{aligned}
\mathfrak{J}_0 &\overset{\text{def}}{=} \{J' \in \mathfrak{J} \mid \text{no } \tau \text{ descendant of } J' \text{ can enable } \overline{a}[A^+]\},\\
\mathfrak{J}_1 &\overset{\text{def}}{=} \{J' \in \mathfrak{J} \mid \text{every } \tau \text{ descendant of } J' \text{ can enable } \overline{a}[A^+] \text{ now or in future}\},\\
\mathfrak{J}_{\frac{1}{2}} &\overset{\text{def}}{=} \mathfrak{J} \setminus (\mathfrak{J}_0 \cup \mathfrak{J}_1).
\end{aligned}
$$

If $J_A \in \mathfrak{J}_0$ then (25) can be bisimulated by $\Omega \,|\, G \xrightarrow{a(A)}= \Omega \,|\, G$ by invoking the component $\lceil !\lambda x.\mathbf{0} \rfloor_a$. Notice that J_A may induce an infinite computation. But this is not a problem in the presence of Ω. If $J_A \in \mathfrak{J}_1$ then (25) can be bisimulated by $\Omega \,|\, G \xrightarrow{a(A)}= \Omega \,|\, G \,|\, \overline{a}[A^+]$ by invoking the component $\lceil !\lambda x.x^+ \rfloor_a$. If $J_A \in \mathfrak{J}_{\frac{1}{2}}$ then (25) can be bisimulated by $\Omega \,|\, G \xrightarrow{a(A)}= \Omega \,|\, G \,|\, (\tau.\overline{a}[A^+] + \tau)$ by invoking the component $\lceil \lambda x.\tau.x^+{+}\tau \rfloor_a$. We only have to prove that every $J' \in \mathfrak{J}_{\frac{1}{2}}$ renders true the following equality.

$$\Omega \,|\, (\tau.\overline{a}[A^+] + \tau) = \Omega \,|\, J'. \tag{26}$$

Suppose $J' \xrightarrow{\tau} J''$. It induces the transition

$$\Omega \,|\, J' \xrightarrow{\tau} \Omega \,|\, J''. \tag{27}$$

If $J'' \in \mathfrak{J}_{\frac{1}{2}}$, then (27) is bisimulated by $\Omega \,|\, (\tau.\overline{a}[A^+] + \tau) \xrightarrow{\tau} \Omega \,|\, (\tau.\overline{a}[A^+] + \tau)$. If $J'' \in \mathfrak{J}_0$, then (27) is bisimulated by $\Omega \,|\, (\tau.\overline{a}[A^+] + \tau) \xrightarrow{\tau} \Omega \,|\, \mathbf{0}$. If $J'' \in \mathfrak{J}_1$, then (27) is bisimulated by $\Omega \,|\, (\tau.\overline{a}[A^+] + \tau) \xrightarrow{\tau} \Omega \,|\, \overline{a}[A^+]$. We are done. ◀

The main result of the paper now follows.

▶ **Theorem 17.** $\mathbb{L} \not\sqsubseteq \pi$.

Theorem 17 provides a seemingly contradictory result to the well-known encoding of higher order π-calculus π^ω in the first order π-calculus [25, 26] and other similar encodings [29, 31]. The higher order process calculi studied in these encodings are both abstraction-passing and complete. The criteria for relative expressiveness adopted in these papers are weaker than the one of this paper. They include extensionality, equipollence and *weak* bisimulation. Some of the encodings satisfy full abstraction property, others satisfy only a weaker form of it. Almost all encodings *op. cit.* satisfy codivergence and branching bisimulation property. As criteria for submodel relationship codivergence and branching bisimulation help to derive reasonable properties about interpretations, which is duly demonstrated in the proof of the general negative result. Without these two criteria we are not able to conclude for example that N_i cannot perform both an input action and an output action.

The source models considered in the above mentioned papers are more powerful than $\mathbb{L}$ in the sense that the latter is restricted by the condition (‡). The equality stated in (13) fails in π^ω since an observer can be powerful enough to detect the presence of the component $\lceil \lambda x.x \overset{?}{>} k \rfloor_a$. Thus Theorem 17 does not contradict to the expressiveness results given by the encodings in the afore mentioned papers, at least on the face of it. We would however like to draw reader's attention to two points. Firstly if we impose the restriction (‡) to π^ω we get a variant, denoted by say $\pi^\omega_\emptyset$, that is comparable to $\mathbb{L}$. The proof of Theorem 17 applies to $\pi^\omega_\emptyset$. It follows that $\pi^\omega_\emptyset \not\sqsubseteq \pi$. On the other hand Sangiorgi's encoding of π^ω in π is also an encoding of $\pi^\omega_\emptyset$ in π. What we do not understand at the moment is if the encoding is still fully abstract. Secondly if we drop the condition (‡) on $\mathbb{L}$, we get a model denoted by $\mathbb{H}$. We may ask the question if $\mathbb{H} \not\sqsubseteq \pi$. Notice that $\mathbb{L} \sqsubseteq \mathbb{H}$ implies $\mathbb{H} \not\sqsubseteq \pi$. Notice also that the failure of (13) implies that the identity map from $\mathbb{L}$ to $\mathbb{H}$ is not a subbisimilarity; it does not rule out however that $\mathbb{L} \sqsubseteq \mathbb{H}$. For the same reason $\pi^\omega_\emptyset \sqsubseteq \pi^\omega$ would imply $\pi^\omega \not\sqsubseteq \pi$. We do not expect that either of the questions, "$\mathbb{L} \sqsubseteq \mathbb{H}$?" and "$\pi^\omega_\emptyset \sqsubseteq \pi^\omega$?", is easy to answer. A possible way to attack these problems is to explore the power of universal processes [6]. This is left for future investigation.

The negative result offered by Theorem 17 provides a fresh look at the issue of higher order calculi vs name-passing calculi and forces us to ask some deeper questions.

6 Conclusion

Name-passing calculi are low level models. It should not come as a surprise that neither the value-passing calculi nor the higher order process calculi with complete messages can be interpreted by the π-calculus. The size of a message in the former models is unbounded. In such models in a single interaction a number/abstraction of arbitrary size is passed from one process to another. This is a high level feature. Studies of communication mechanisms at different abstract levels are part of process theory.

Let's summarize what we know about the relative expressiveness of the three classes of model. Now "Name-Passing $\not\sqsubseteq$ Value-Passing $\not\sqsubseteq$ Name-Passing" [5] and for what we know at the moment "Process-Passing $\not\sqsubseteq$ Name-Passing". It is easily seen that "Process-Passing $\not\sqsubseteq$ Value-Passing" because value-passing communications cannot interpret name extrusion. The problem "Value-Passing $\sqsubseteq$ Process-Passing?" deserves attention. The numbers can be coded up by abstractions. The question is how to do it in a bijective way.

Having seen all the negative results, one wonders if there is a communication mechanism that can interpret the name-passing, the value-passing and the abstraction-passing mechanisms. In [7] a universal model $\mathbb{V}$ is defined that can indeed interpret for example the model $\mathbb{L}$, the π-calculus and the value-passing calculus [3]. Such a model, rather than being artificial, is well motivated by the Church-Turing Thesis. It is against this overall picture that the result of this short paper is to be appreciated.

Acknowledgments. We thank the anonymous reviewers for their criticisms and comments. We thank Davide Sangiorgi for his expert advices. We thank BASICS members for their interest.

References

1 S. Abramsky. The Lazy Lambda Calculus. In: Turner, D. Ed., *Declarative Programming.* Addison-Wesley, 65-116, 1988.

2 H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* North-Holland, 1984.

3 Y. Fu. The Value-Passing Calculus. In: *Theories of Programming and Formal Methods*, Lecture Notes in Computer Science 8051, 166-195, 2013.

4 Y. Fu. Nondeterministic Structure of Computation. *Mathematical Structures in Computer Science*, 25:1295-1338, 2015.

5 Y. Fu. Theory of Interaction. *Theoretical Computer Science*, 611:1-49, 2016.

6 Y. Fu. The Universal Process. To appear in *Logical Methods in Computer Science*, 2017.

7 Y. Fu. Thesis for Interaction. `http://basics.sjtu.edu.cn/ yuxi/`. 2017.

8 Y. Fu and H. Lu. On the Expressiveness of Interaction. *Theoretical Computer Science*, 411:1387-1451, 2010.

9 Y. Fu and H. Zhu. The Name-Passing Calculus. arXiv:1508.00093, 2015.

10 D. Gorla. Towards a Unified Approach to Encodability and Separation Results for Process Calculi. In: *CONCUR'08*, Lecture Notes in Computer Science 5201, 492-507, 2008.

11 M. Hennessy and A. Ingólfsdóttir. A Theory of Communicating Processes with Value-Passing. *Information and Computation*, 107:202-236, 1993.

12 M. Hennessy and H. Lin. Symbolic Bisimulations. *Theoretical Computer Science*, 138:353-369, 1995.

13 C. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

14 R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

15 R. Milner. Functions as Processes. *Mathematical Structures in Computer Science*, 2:119-146, 1992.

16 R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes. *Information and Computation*, 100:1-40 (Part I), 41-77 (Part II), 1992.

17 R. Milner and D. Sangiorgi. Barbed Bisimulation. In: *ICALP'92*, Lecture Notes in Computer Science 623, 685-695, 1992.

18 U. Nestmann. What is a Good Encoding of Guarded Choices? *Information and Computation*, 156:287-319, 2000.

19 U. Nestmann. Welcome to the Jungle: A Subjective Guide to Mobile Process Calculi. In: *CONCUR'06*, Lecture Notes in Computer Science 4137, 52-63, 2006.

20 C. Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous π-Calculus. *Mathematical Structures in Computer Science*, 13:685-719, 2003.

21 D. Park. Concurrency and Automata on Infinite Sequences. In: *Theoretical Computer Science*, Lecture Notes in Computer Science 104, 167-183, 1981.

22 J. Parrow. An Introduction to the π-Calculus. In: J. Bergstra, A. Ponse and S. Smolka (Eds.), *Handbook of Process Algebra*. North-Holland, 478-543, 2001.

23 J. Parrow. Expressiveness of Process Algebras. In: *LIX Colloquium'06*, 2006.

24 L. Priese. On the Concept of Simulation in Asynchronous, Concurrent Systems. *Progress in Cybernatics and Systems Research*, 7:85-92, 1978.

25 D. Sangiorgi. Expressing Mobility in Process Algebras: First Order and Higher Order Paradigm. Ph.D. thesis, Department of Computer Science, University of Edinburgh, 1992.

26 D. Sangiorgi. From π-Calculus to Higher Order π-Calculus – and Back. In: *TAPSOFT'93*, Lecture Notes in Computer Science 668, 151-166, 1993.

27 D. Sangiorgi. π-Calculus, Internal Mobility and Agent-Passing Calculi. *Theoretical Computer Science*, 167:235-274, 1996.

28 D. Sangiorgi and D. Walker. *The π Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

29 B. Thomsen. A Calculus of Higher Order Communicating Systems. In: *POPL'89*, 143-154, 1989.

30 B. Thomsen. A Theory of Higher Order Communicating Systems. *Information and Computation*, 116:38-57, 1995.

31 X. Xu. Distinguishing and Relating Higher-Order and First-Order Processes by Expressiveness. *Acta Informatica*, 49:445-484, 2012.

32 X. Xu and Q. Yin and H. Long: On the Computation Power of Name Parameterization in Higher-Order Processes. In: *ICE'15*, 114-127, 2015.

33 R. van Glabbeek and W. Weijland. Branching Time and Abstraction in Bisimulation Semantics. In: *Information Processing'89*, 613-618, 1989.

34 D. Walker. Objects in the π-Calculus. *Information and Computation*, 116:253-271, 1995.