From Ambients To A Routing Calculus

Xudong Guan [1,2]

INRIA, 2004 Route des Lucioles, 06902 Sophia Antipolis, France

Abstract

From a more simplified encoding of π-calculus into pure ambients, we derive a routing calculus which is simple and expressive. While being a direct subset of pure robust ambients, the basic version of this calculus is able to encode π, and simulate Dπ to some extent. By adding a primitive that enables mobile agent acquiring the current location name, the calculus is able to give a reasonable and much simpler encoding of Dπ.

1 Introduction

The π-calculus [MPW92,SW01] is a formal model for concurrent and mobile systems based on name-passing between parallel processes. The calculus of mobile ambients [CG00] is a formal model for mobile and distributed computation, where computation is modeled by ambients moving around and being opened. Recently, results have been shown [Zim00] that name-passing in π-calculus can be simulated by ambient movements and openings. This paper presents our further investigation into this problem.

Our start point is a new encoding from π to pure [3] robust ambients[GYY00] (ROAM). The encoding is simpler, in terms of both size and the computation steps required to simulate one π-reduction, than Zimmer's encoding and our previous one [Gua02]. Like our previous one, this new encoding is a well-behaved one, in that ambients are either single-threaded or immobile. From this encoding, we design a sub-calculus of ROAM called $\texttt{PR}^-$ that allows only single-threaded or immobile ambients by syntactical means, while still being able to encode π. Meanwhile, $\texttt{PR}^-$ itself exhibits its own packet-routing model for mobile and distributed computation.

[1] Supported by EU project 'MIKADO: Mobile Calculi based on Domains', FET-GC IST-2001-32222.
[2] Email: `Xudong.Guan@sophia.inria.fr`
[3] In ambient jargon, *pure* stands for the basic version without communication.

As a step further on the expressiveness of $\texttt{PR}^-$ we present an encoding of Dπ [HR98], a distributed π-calculus supporting process distribution, migration, and dynamic-binding of channels. In the encoding, we show how to realize both channel name substitution and location name substitution in ambients. As one can expect, the encoding is rather complex, even though it is written in the more readable $\texttt{PR}^-$ instead of ROAM.

To overcome this problem, we add to $\texttt{PR}^-$ a primitive called *out-binding* that enables ambients to know the name of their locations. For simplicity, we also anonymize all single-threaded ambients. In this new calculus, the encoding of π and Dπ are simpler. Although this calculus is no longer a proper sub-calculus of ROAM, it is also interesting and worth further exploration in its own.

The rest of this paper is organized as follows. Section 2 gives a review of π and ROAM. Section 3 presents the more concise encoding. Section 4 gives the corresponding encoding in $\texttt{PR}^-$, and gives a translation of $\texttt{PR}^-$ processes to ROAM processes. Section 5 studies the encoding of Dπ in $\texttt{PR}^-$. Section 6 presents the extension and the corresponding new encodings in the extended calculus. Finally, Section 7 concludes the paper. In the appendix, we list the two previous encodings from π to pure ambients, and give some comparisons of them to the one proposed in this paper.

2 Review of the two languages

2.1 π-calculus

As in [Zim00], we use the sum-free synchronous π-calculus (π-calculus for short) as our source language. For the sake of encoding, names are distinguished by two non-overlapping sets, the set of channel names (often ranged over by $n, m, ...$), and the set of variable names (often ranged over by $x, y, ...$), collectively called values (denoted by $u, v, ...$). The set of π-processes is defined by the follow grammar:

$$P, Q \quad ::= \quad \mathbf{0} \quad | \quad P \mid Q \quad | \quad !\,P \quad | \quad (\nu n)P \quad | \quad u\langle v\rangle.P \quad | \quad u(x).P$$

The main reduction rule of π-calculus is the following communication rule, enabling an *output* process on channel n ($n\langle u\rangle.P$) to pass a name to an input process on the same channel ($n(x).P$).

$$n\langle u\rangle.P \mid n(x).Q \longrightarrow P \mid Q\{x{:=}u\}$$

Reductions can happen through structural re-arrangements, and inside evaluation contexts. The evaluation context of π is defined as:

$$E \quad ::= \quad - \quad | \quad E \mid P \quad | \quad (\nu n)E$$

More detailed explanation could be found in standard π-calculus literature [MPW92,SW01].

2.2 Robust ambients and type system

ROAM is a variant of safe ambients [LS00] (SA), having slightly different semantics for co-actions. We briefly review a typed version of ROAM in this section. More details could be found in [GYY00,GYY01,Gua02].

Given a set of names range over by $n, m, ...$, the set of ROAM processes is defined as:

$$P ::= \mathbf{0} \quad | \quad P \mid Q \quad | \quad !\,P \quad | \quad (\nu n : T)P \quad | \quad n[P] \quad | \quad M.P$$

Parallel composition, replication and restriction have their usually meanings. $n[P]$ stands for an *ambient* able to perform different *actions* according to the running process P inside. $M.P$ is a process that can perform action M and then becomes P. There are six different types of actions in ROAM, formulated in three pairs [4].

$$M ::= \mathtt{in}\ n \quad | \quad \overline{\mathtt{in}}\ n \quad | \quad \mathtt{out} \quad | \quad \overline{\mathtt{out}}\ n \quad | \quad \mathtt{open}\ n \quad | \quad \overline{\mathtt{open}}$$

Actions, except $\mathtt{open}\ n$, command their surrounding ambients. We introduce the functionalities of these actions briefly: $\mathtt{in}\ n$ tells the surrounding ambient to move inside ambient n, while $\overline{\mathtt{in}}\ n$ allows an ambient named n to move inside; $\mathtt{out}$ tells the surrounding ambient to move outside its current location, while $\overline{\mathtt{out}}\ n$ allows an ambient named n to go out; $\mathtt{open}\ n$ opens ambient n, while $\overline{\mathtt{open}}$ gives permission for others to open the ambient. The evaluation contexts and reduction axioms of ROAM is summarized below. Reduction can not only happen inside compositions and restrictions, but also inside ambients.

$$\begin{array}{rcl} E & ::= & - \quad | \quad E \mid P \quad | \quad (\nu n : T)E \quad | \quad n[E] \\ m[\mathtt{in}\ n.P_1 \mid P_2] \mid n[\overline{\mathtt{in}}\ m.Q_1 \mid Q_2] & \longrightarrow & n[m[P_1 \mid P_2] \mid Q_1 \mid Q_2] \\ n[m[\mathtt{out}.P_1 \mid P_2] \mid \overline{\mathtt{out}}\ m.Q_1 \mid Q_2] & \longrightarrow & m[P_1 \mid P_2] \mid n[Q_1 \mid Q_2] \\ \mathtt{open}\ n.P \mid n[\overline{\mathtt{open}}.Q_1 \mid Q_2] & \longrightarrow & P \mid Q_1 \mid Q_2 \end{array}$$

Every ambient is assigned a type T. It can be a *simple type* Z^Y, having mobility Z ($\curvearrowright$ - mobile, or $\underline{\vee}$ - immobile) and threads Y (0 - no thread, 1 - single-threaded, or ω - multi-threaded), and can't be opened. It can also be an *evolving type* $Z^Y[T]$, denoting that the ambient, apart from having mobility Z and threads Y, can be opened and release processes of type [5] T. We are mainly interested in *well-behaved* processes in which ambients are either immobile or single-threaded. The results in [Gua02] shows that well-behaved processes have a better behavior theory than arbitrary processes.

---

[4] We use a version of ROAM with no parameter for action "$\mathtt{out}$".

[5] Ambient type is actually computed from the type of the process inside it. Refer to [GYY01] for more details.

$$
\begin{aligned}
\mathbf{fwd}\ u &\triangleq\ !\,route[\overline{\mathtt{in}}\ route.\mathtt{open}\ route.\mathtt{out}.\mathtt{in}\ u.\mathtt{in}\ route.\overline{\mathtt{open}}]\\
\langle\!\langle P\rangle\!\rangle^{*} &\triangleq \langle\!\langle P\rangle\!\rangle \mid !\,\mathtt{open}\ cont\\
\langle\!\langle \mathbf{0}\rangle\!\rangle &\triangleq \mathbf{0}\\
\langle\!\langle P \mid Q\rangle\!\rangle &\triangleq \langle\!\langle P\rangle\!\rangle \mid \langle\!\langle Q\rangle\!\rangle\\
\langle\!\langle\, !\,P\rangle\!\rangle &\triangleq\ !\,\langle\!\langle P\rangle\!\rangle\\
\langle\!\langle (\nu n)P\rangle\!\rangle &\triangleq (\nu n : \underline{\vee}^{\omega})\ (\ n[\ !\,\overline{\mathtt{in}}\ route \mid !\,\mathtt{open}\ route \mid !\,\overline{\mathtt{out}}\ comm\\
&\qquad\qquad \mid !\,route[\overline{\mathtt{in}}\ route.\mathtt{open}\ route.\overline{\mathtt{open}}]\ ] \quad\mid\quad \langle\!\langle P\rangle\!\rangle)\\
\langle\!\langle u\langle v\rangle.P\rangle\!\rangle &\triangleq route\ [\ \mathtt{in}\ u.\mathtt{in}\ route.\overline{\mathtt{open}}\\
&\qquad \mid\ comm\ [\ \mathtt{in}\ comm.\overline{\mathtt{open}} \quad\mid\quad \mathbf{fwd}\ v\\
&\qquad\qquad \mid\ cont[\mathtt{out}.\overline{\mathtt{open}} \mid \langle\!\langle P\rangle\!\rangle]]]\\
\langle\!\langle u(x).P\rangle\!\rangle &\triangleq (\nu x : \underline{\vee}^{\omega})\ (\ route[\ \mathtt{in}\ u.\mathtt{in}\ route.\overline{\mathtt{open}}\\
&\qquad\qquad \mid\ comm[\ \overline{\mathtt{in}}\ comm.\mathtt{open}\ comm.\mathtt{out}.\mathtt{in}\ x.\overline{\mathtt{open}}\\
&\qquad\qquad\qquad \mid\ cont[\mathtt{out}.\overline{\mathtt{open}} \mid \langle\!\langle P\rangle\!\rangle]]]\\
&\qquad \mid\ x[\ !\,\overline{\mathtt{in}}\ route \mid !\,\overline{\mathtt{out}}\ route\\
&\qquad\quad \mid !\,\overline{\mathtt{out}}\ cont \mid !\,\overline{\mathtt{in}}\ comm \mid !\,\mathtt{open}\ comm])
\end{aligned}
$$

Fig. 1. Encoding π-calculus in ROAM: a concise one.

3 A more concise encoding

As our start point, we give in this section a new encoding of π-calculus into well-behaved ROAM processes. We write the encoding $\langle\!\langle P\rangle\!\rangle^{*}$. We use three special ambient names: *route* for routing communication processes, *comm* for interaction between communication processes, and *cont* for wrapping continuation processes. The encoding is defined only for closed processes without free names. For those having free names, we need to supply additional free channel ambients in parallel to the encoding.

The encoding works as follows. For every channel, we construct an immobile ambient with the same name, inside which input and output processes can meet and communicate. An input/output process has the form $route[\mathtt{in}\ u.\mathtt{in}\ route.\overline{\mathtt{open}} \mid P]$. It can route its cargo P, composed by all the sub-ambients of it, to the correct channel specified by u, which is either a channel name itself, or a variable name. The routing service in a channel ambient, $!\,route[\overline{\mathtt{in}}\ route.\mathtt{open}\ route.\overline{\mathtt{open}}]$, tells the incoming packet that this is the right destination. Meanwhile, the routing service **fwd** v in a variable ambient turns the incoming packet to the one having destination v.

Let's now see how read and write process actually interact when they reach the right channel. The cargo of a read process with the variable x and continuation P is of the form

$$comm[\overline{\mathtt{in}}\ comm.\mathtt{open}\ comm.\mathtt{out}.\mathtt{in}\ x.\overline{\mathtt{open}} \mid cont[\mathtt{out}.\overline{\mathtt{open}} \mid \langle\!\langle P\rangle\!\rangle]],$$

while the cargo of a write process sending v with continuation Q is:

$$comm[\mathtt{in}\ comm.\overline{\mathtt{open}} \mid \mathbf{fwd}\ v \mid cont[\mathtt{out}.\overline{\mathtt{open}} \mid \langle\!\langle Q\rangle\!\rangle]]$$

$$
\begin{array}{rcl}
\mathbf{fwd}\; u & \triangleq & !\,\mathit{route}(\texttt{get}\;\mathit{route}.\texttt{out}.\texttt{in}\;u.\texttt{put}\;\mathit{route})[\,] \\
\langle\!\langle P\rangle\!\rangle^{*} & \triangleq & \langle\!\langle P\rangle\!\rangle \mid \mathit{cont}^{-} \\
\langle\!\langle \mathbf{0}\rangle\!\rangle & \triangleq & \mathbf{0} \\
\langle\!\langle P \mid Q\rangle\!\rangle & \triangleq & \langle\!\langle P\rangle\!\rangle \mid \langle\!\langle Q\rangle\!\rangle \\
\langle\!\langle\, !\,P\rangle\!\rangle & \triangleq & !\,\langle\!\langle P\rangle\!\rangle \\
\langle\!\langle (\nu n)P\rangle\!\rangle & \triangleq & (\nu n)(\; n\,\langle \mathit{route}^{\ddagger} \mid \mathit{comm}^{\uparrow}\rangle \\
& & \qquad\qquad [\;!\,\mathit{route}(\texttt{get}\;\mathit{route}.\texttt{dis})[\,]\;] \quad\mid\quad \langle\!\langle P\rangle\!\rangle) \\
\langle\!\langle u\langle v\rangle.P\rangle\!\rangle & \triangleq & \mathit{route}\;(\;\texttt{in}\;u.\texttt{put}\;\mathit{route}) \\
& & \qquad [\;\mathit{comm}\;(\;\texttt{put}\;\mathit{comm}) \\
& & \qquad\qquad [\;\mathbf{fwd}\;v \mid\;\; \mathit{cont}(\texttt{out}.\texttt{dis})[\langle\!\langle P\rangle\!\rangle]\;]\;\;] \\
\langle\!\langle u(x).P\rangle\!\rangle & \triangleq & (\nu x)\;(\;\mathit{route}(\;\texttt{in}\;u.\texttt{put}\;\mathit{route}) \\
& & \qquad\qquad [\;\mathit{comm}(\texttt{get}\;\mathit{comm}.\texttt{out}.\texttt{in}\;x.\texttt{dis}) \\
& & \qquad\qquad\qquad [\;\mathit{cont}(\texttt{out}.\texttt{dis})[\langle\!\langle P\rangle\!\rangle]\;]\;\;] \\
& & \qquad \mid\;\; x\,\langle \mathit{route}^{\updownarrow} \mid \mathit{cont}^{\uparrow} \mid \mathit{comm}^{\ddagger}\rangle[\,]\;)
\end{array}
$$

Fig. 2. Encoding π-calculus in ROAM using abbreviations

The result of their interaction is an ambient that first come out of the channel ambient, then go into the variable ambient x with the redirector $\mathbf{fwd}\; v$, together with two continuation processes P and Q, which are then released after communication.

Compared with previous encodings in SA and ROAM, this one works under the same framework, but is simpler in terms of both encoding size and computation steps required to simulate one π-reduction. See Appendix B for detailed comparisons of these encodings. The encoding is also well-behaved, in that we have all the channel ambients and variable ambients being immobile, and the rest being single-threaded (all the three system ambients have type $\curvearrowright^{1} [\underline{\vee}^{0}]$).

4 Rewrite the encoding using abbreviations

In this section, we present a new syntax for a subset of well-behaved ROAM processes. We let $\texttt{get}\; a.P \triangleq \overline{\texttt{in}}\; a.\texttt{open}\; a.P$, $\texttt{put}\; a \triangleq \texttt{in}\; a.\overline{\texttt{open}}.\mathbf{0}$, and $\texttt{dis} \triangleq \overline{\texttt{open}}.\mathbf{0}$. We write $a(M)[P]$ for single-threaded ambients $a[M.\mathbf{0} \mid P]$, where P is the collection of all sub-ambients of a. We use abbreviations like $a^{\downarrow} \triangleq !\,\overline{\texttt{in}}\; a$, $a^{\uparrow} \triangleq !\,\overline{\texttt{out}}\; a$, and $a^{-} \triangleq !\,\texttt{open}\; a$. We write $a^{\ddagger}$ for $a^{\downarrow} \mid a^{\uparrow} \mid a^{-}$, etc. We write $a\langle C\rangle[P]$ for immobile ambients $a[C \mid P]$, where C are those three kinds of replicated actions. We omit the type annotations in Figure 1 and re-write the encoding using these abbreviations in Figure 2.

The routing protocol in Figure 1 can be expressed concretely using abbreviations like *get* and *put*. To go a step further, we define a subset of ROAM processes using these notations so that all processes in the subset are well-

$$
\begin{array}{llll}
\text{Processes:} & & \text{Names:} & \\
P, Q ::= \mathbf{0} & \text{empty} & n, m ::= w & \text{wall name} \\
\quad \mid P \mid Q & \text{composition} & \quad \mid a & \text{agent name} \\
\quad \mid\ !\,P & \text{replication} & & \\
\quad \mid (\nu n)P & \text{restriction} & & \\
\quad \mid a(M)[P] & \text{single threaded ambient} & & \\
\quad \mid w\langle C\rangle[P] & \text{immobile ambient} & &
\end{array}
$$

$$
\begin{array}{llll}
\text{Capabilities:} & & \text{Controllers:} & \\
M, N ::= \texttt{in}\ w.M & \text{move in} & C, D ::= \mathbf{0} & \text{empty} \\
\quad \mid \texttt{out}.M & \text{move out} & \quad \mid a^{\downarrow} & \text{allow move in} \\
\quad \mid \texttt{get}\ a.M & \text{get some cargo} & \quad \mid a^{\uparrow} & \text{allow move out} \\
\quad \mid \texttt{put}\ a & \text{put cargo} & \quad \mid a^{-} & \text{allow dissolving} \\
\quad \mid \texttt{dis} & \text{dissolve} & \quad \mid C \mid C' & \text{composition}
\end{array}
$$

Evaluation contexts:

$$E \quad ::= \quad - \quad \mid \quad E \mid P \quad \mid \quad (\nu n)E \quad \mid \quad a(M)[E] \quad \mid \quad w\langle C\rangle[E]$$

Reduction axioms:

$$
\begin{array}{rcl}
w\langle a^{\downarrow} \mid C\rangle[P] \mid a(\texttt{in}\ w.M)[Q] & \longrightarrow & w\langle a^{\downarrow} \mid C\rangle[P \mid a(M)[Q]] \\
w\langle a^{\uparrow} \mid C\rangle[P \mid a(\texttt{out}.M)[Q]] & \longrightarrow & w\langle a^{\uparrow} \mid C\rangle[P] \mid a(M)[Q] \\
w\langle a^{-} \mid C\rangle[P \mid a(\texttt{dis})[Q]] & \longrightarrow & w\langle a^{-} \mid C\rangle[P \mid Q] \\
a(\texttt{get}\ b.M)[P] \mid b(\texttt{put}\ a)[Q] & \longrightarrow & a(M)[P \mid Q]
\end{array}
$$

Fig. 3. Syntax and reduction semantics of $\texttt{PR}^{-}$

$$
\begin{array}{rcl@{\qquad}rcl}
\{\![\texttt{in}\ w.M]\!\} & \triangleq & \texttt{in}\ w.\{\![M]\!\} & \{\![\mathbf{0}]\!\} & \triangleq & \mathbf{0} \\
\{\![\texttt{out}.M]\!\} & \triangleq & \texttt{out}.\{\![M]\!\} & \{\![P \mid Q]\!\} & \triangleq & \{\![P]\!\} \mid \{\![Q]\!\} \\
\{\![\texttt{get}\ a.M]\!\} & \triangleq & \overline{\texttt{in}}\ a.\texttt{open}\ a.\{\![M]\!\} & \{\![\,!\,P]\!\} & \triangleq & !\,\{\![P]\!\} \\
\{\![\texttt{put}\ a]\!\} & \triangleq & \texttt{in}\ a.\overline{\texttt{open}}.\mathbf{0} & \{\![(\nu a)P]\!\} & \triangleq & (\nu a : \curvearrowright^{1} [\,\underline{\vee}^{\,0}])\{\![P]\!\} \\
\{\![\texttt{dis}]\!\} & \triangleq & \overline{\texttt{open}}.\mathbf{0} & \{\![(\nu w)P]\!\} & \triangleq & (\nu w : \underline{\vee}^{\,\omega})\{\![P]\!\} \\
\{\![a^{\downarrow}]\!\} & \triangleq & !\,\overline{\texttt{in}}\ a & \{\![a(M)[P]]\!\} & \triangleq & a[\{\![M]\!\} \mid \{\![P]\!\}] \\
\{\![a^{\uparrow}]\!\} & \triangleq & !\,\overline{\texttt{out}}\ a & \{\![w\langle C\rangle[P]]\!\} & \triangleq & w[\{\![C]\!\} \mid \{\![P]\!\}] \\
\{\![a^{-}]\!\} & \triangleq & !\,\texttt{open}\ a & & & \\
\{\![C \mid D]\!\} & \triangleq & \{\![C]\!\} \mid \{\![D]\!\} & & &
\end{array}
$$

Fig. 4. Translating $\texttt{PR}^{-}$ to ROAM

behaved. We separate the set of ambient names to agent names (ranged over by $a, b, \ldots$) and wall names (ranged over by $w, v, \ldots$), and define *pure ROAM minus* ($\texttt{PR}^{-}$), a sub-calculus of ROAM in Figure 3. The reduction semantics is defined to be compliant with that of ROAM.

Clearly, every $\texttt{PR}^{-}$ process is also a ROAM process. Translating a $\texttt{PR}^{-}$ process P back to a ROAM process, $\{\![P]\!\}$, can be defined easily in Figure 4.

We have the following result on the translation.

Networks:

$$
\begin{array}{rcll}
N, M & ::= & \mathbf{0} & \text{empty} \\
& \mid & M \mid N & \text{composition} \\
& \mid & (\nu l)N & \text{location res.} \\
& \mid & (\nu_l\, a)N & \text{channel res.} \\
& \mid & l[\![P]\!] & \text{location}
\end{array}
$$

$$
\begin{array}{lll}
u_{loc} ::= l \mid x_{loc} & u ::= u_{loc} & \mid\ u_{chn} \\
u_{chn} ::= a \mid x_{chn} & x ::= x_{loc} & \mid\ x_{chn}
\end{array}
$$

Processes:

$$
\begin{array}{rcll}
P, Q & ::= & \texttt{stop} & \text{empty} \\
& \mid & P \mid Q & \text{composition} \\
& \mid & !\,P & \text{replication} \\
& \mid & (\nu l)P & \text{location res.} \\
& \mid & (\nu a)P & \text{channel res.} \\
& \mid & \texttt{go}\ u_{loc}.P & \text{migration} \\
& \mid & u_{chn}\langle u\rangle.P & \text{write value} \\
& \mid & u_{chn}(x).P & \text{read value}
\end{array}
$$

Evaluation contexts:

$$
E \quad ::= \quad - \quad \mid \quad E \mid N \quad \mid \quad (\nu l)N \quad \mid \quad (\nu_l\, a)E
$$

Reduction axioms:

$$
\begin{array}{rcl}
l[\![P \mid Q]\!] & \longrightarrow & l[\![P]\!] \mid l[\![Q]\!] \\
l[\![\texttt{go}\ k.P]\!] & \longrightarrow & k[\![P]\!] \\
l[\![a\langle u_{loc}\rangle.P]\!] \mid l[\![a(x_{loc}).Q]\!] & \longrightarrow & l[\![P]\!] \mid l[\![Q\{x_{loc} := u_{loc}\}]\!] \\
l[\![a\langle u_{chn}\rangle.P]\!] \mid l[\![a(x_{chn}).Q]\!] & \longrightarrow & l[\![P]\!] \mid l[\![Q\{x_{chn} := u_{chn}\}]\!]
\end{array}
$$

Fig. 5. Syntax and reduction semantics of Dπ

Theorem 4.1 *For any* $\texttt{PR}^-$ *process* P, $\{\![P]\!\}$ *is a well-behaved ROAM process.*

Proof. This can be proved easily according to definition of $\{\![P]\!\}$. Every agent ambient can be assigned with type $\curvearrowright^1 [\underline{\vee}^0]$. Every wall ambient can be assigned with type $\underline{\vee}^\omega$. $\square$

Albeit a sub-calculus of ROAM, $\texttt{PR}^-$ also exhibits a packet-routing model at an higher abstraction level than ROAM. It makes ROAM processes more readable and makes it easier to write complex processes. In the next section, we show an encoding of Dπ in ambients written in $\texttt{PR}^-$.

5 The encoding of Dπ

The calculus of Dπ [HR98] adds distributed location on top of the π-calculus. Processes are located and written $l[\![P]\!]$. A new primitive **go** $l.P$ can migrate process P to location l. Communication can only happen if both the read process and the write process reside at the same location. During communication, either channel names or location names can be communicated.

For the sake of encoding, we distinguish channels from locations. We let l, k range over location names, a, b range over channel names, x_{loc}, y_{loc} range over location variables, and x_{chn}, y_{chn} range over channel variables. The syntax and reduction semantics of Dπ is summarized in Figure 5.

To simulate Dπ, we use two layers of walls: a top layer for Dπ locations, and a second layer for Dπ channels. To realize substitution, we need to build, as what we have done in the encoding of π, redirection ambients $x[\mathbf{fwd}\ u]$ that can route communication ambients *comm* to the right channel. In Dπ, both channels and locations can be communicated. We identify 4 kinds of redirection ambients as follows:

(i) $x_{loc}[\mathbf{fwd}(l)]$: location variable x_{loc} instantiated as location name l;

(ii) $x_{loc}[\mathbf{fwd}(y_{loc})]$: location variable x_{loc} instantiated as another location variable y_{loc};

(iii) $x_{chn}[\mathbf{fwd}(a)]$: channel variable x_{chn} instantiated as channel name a;

(iv) $x_{chn}[\mathbf{fwd}(y_{chn})]$: channel variable x_{chn} instantiated as another channel variable y_{chn}.

We need to decide which layer should those redirection ambients be located. Those redirection ambients for location variables should be parallel with locations, so that it makes little different between processes migrating to locations and processes migrating to redirection ambients. As for redirection ambients for channel variables, we put them also in parallel with locations, since otherwise, if we put them in parallel with channels, we need to create one such redirecting ambients in every location, even in newly created ones.

One interesting property in Dπ is the ability to dynamically bind channel names. For example, in process $b(x).\texttt{go}\ x.a(y).P$, the location of channel a is not known in advance. The actual channel that a will interact with depends on the value it receives from b. To encode this process using redirection ambients, the process $a(y).P$ must find its way back to where it from after interaction with the redirection ambient y outside of its location. We need some tricks in the encoding to enable processes come back to where they were. For this reason, we add two primitives in $\texttt{PR}^-$: "$\texttt{wait}\ a$" and "$\texttt{sig}\ a$". They enable an agent to "wait" for some activities of its cargo before it fires the dissolve action. Their definitions and operational semantics are summarized below.

$$M ::= \ldots \quad | \quad \texttt{wait}\ a.M \quad | \quad \texttt{sig}\ a.M$$

$$a(\texttt{wait}\ s.M)[P \mid b(\texttt{sig}\ s.N)[Q]] \longrightarrow a(M)[P \mid b(N)[Q]]$$

Their corresponding implementation in ROAM are defined as:

$$\{[\texttt{wait}\ s.M]\} \triangleq \overline{\texttt{out}}\ s.\texttt{open}\ s.\{[M]\}$$

$$\{[\texttt{sig}\ s.M]\} \triangleq s[\texttt{out}.\overline{\texttt{open}}] \mid \{[M]\}$$

It is easy to verify that Theorem 4.1 in the previous section still holds with these extensions, and $\texttt{PR}^-$ with these addition is still a sub-calculus of ROAM.

Now we can write the encoding from Dπ network to $\texttt{PR}^-$ processes: $\{\!\{N\}\!\}^*$.

$$\{\!\{N\}\!\}^* \triangleq root\langle tonet^-\rangle[\{\!\{N\}\!\}]$$

$$\begin{aligned}
\{\!\{\mathbf{0}\}\!\} &\triangleq \mathbf{0} \\
\{\!\{N \mid M\}\!\} &\triangleq \{\!\{N\}\!\} \mid \{\!\{M\}\!\} \\
\{\!\{(\nu l)N\}\!\} &\triangleq (\nu l)(l\langle \mathbf{C_l}\rangle[\mathbf{B}(l)] \mid \{\!\{N\}\!\}) \\
\{\!\{(\nu_l a)N\}\!\} &\triangleq (\nu a)(toloc(\texttt{in}\ l.\texttt{dis})[a\langle \mathbf{C_n}\rangle[\,]] \mid \{\!\{N\}\!\}) \\
\{\!\{l[\![P]\!]\}\!\} &\triangleq toloc(\texttt{in}\ l.\texttt{dis})[\{\!\{P\}\!\}]
\end{aligned}$$

$$
\begin{aligned}
\{\!\{\texttt{stop}\}\!\} &\triangleq \mathbf{0} \\
\{\!\{P \mid Q\}\!\} &\triangleq \{\!\{P\}\!\} \mid \{\!\{Q\}\!\} \\
\{\!\{\,!\,P\}\!\} &\triangleq \,!\,\{\!\{P\}\!\} \\
\{\!\{(\nu l)P\}\!\} &\triangleq (\nu l)(\mathit{tonet}(\texttt{out.dis})[l\langle \mathbf{C_l}\rangle[\mathbf{B}(l)]] \mid \{\!\{P\}\!\}) \\
\{\!\{(\nu a)P\}\!\} &\triangleq (\nu a)(a\langle \mathbf{C_n}\rangle[\,] \mid \{\!\{P\}\!\}) \\
\{\!\{\texttt{go}\ l.P\}\!\} &\triangleq \mathit{toloc}(\texttt{out.in}\ l.\texttt{dis})[\{\!\{P\}\!\}] \\
\{\!\{\texttt{go}\ x_{loc}.P\}\!\} &\triangleq \mathit{toloc}(\texttt{out.in}\ x_{loc}.\texttt{put}\ \mathit{toloc})[\{\!\{P\}\!\}] \\
\{\!\{a\langle u\rangle.P\}\!\} &\triangleq \mathit{tochn}(\texttt{in}\ a.\texttt{dis})[\mathbf{O}(u,P)] \\
\{\!\{x_{chn}\langle u\rangle.P\}\!\} &\triangleq \mathit{tovar}(\texttt{get}\ \mathit{back}.\texttt{out.in}\ x_{chn}.\texttt{put}\ \mathit{tovar}) \\
&\qquad\qquad [\,\mathit{toloc}(\texttt{put}\ \mathit{toloc})[\,\mathit{tochn}(\texttt{put}\ \mathit{tochn})[\mathbf{O}(u,P)]\,]\,] \\
\{\!\{a(x).P\}\!\} &\triangleq (\nu x)\ (\ \mathit{tonet}(\texttt{out.dis})[\mathbf{Var}(x)] \\
&\qquad\qquad \mid \mathit{tochn}(\texttt{in}\ a.\texttt{dis})[\mathbf{I}(x,P)]\ ) \\
\{\!\{x_{chn}(x).P\}\!\} &\triangleq (\nu x)\ (\ \mathit{tonet}(\texttt{out.dis})[\mathbf{Var}(x)] \\
&\qquad\qquad \mid \mathit{tovar}(\texttt{get}\ \mathit{back}.\texttt{out.in}\ x_{chn}.\texttt{put}\ \mathit{tovar}) \\
&\qquad\qquad\qquad [\mathit{toloc}(\texttt{put}\ \mathit{toloc})[\,\mathit{tochn}(\texttt{put}\ \mathit{tochn})[\mathbf{I}(x,P)]]]])
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{O}(u,P) &\triangleq \mathit{comm}(\texttt{put}\ \mathit{comm}) \\
&\qquad\qquad [\,\mathit{tovar}(\texttt{put}\ \mathit{tovar})[\mathbf{fwd}(u)] \mid \mathit{cont}(\texttt{dis})[\{\!\{P\}\!\}]] \\
\mathbf{I}(x,P) &\triangleq \mathit{comm}(\texttt{get}\ \mathit{comm}.\texttt{out.wait}\ s.\texttt{dis}) \\
&\qquad\qquad [\,\mathit{tovar}(\texttt{get}\ \mathit{tovar}.\texttt{sig}\ s.\texttt{out.in}\ x.\texttt{dis})[\,] \\
&\qquad\qquad \mid\ \mathit{cont}(\texttt{dis})[\{\!\{P\}\!\}]\,] \\
\mathbf{fwd}(a) &\triangleq \,!\,\mathit{tovar}(\texttt{get}\ \mathit{tovar}.\texttt{wait}\ s.\texttt{dis}) \\
&\qquad\qquad [\mathit{toloc}(\texttt{put}\ \mathit{toloc})[\mathit{tochn}(\texttt{get}\ \mathit{tochn}.\texttt{sig}\ s.\texttt{in}\ a.\texttt{dis})[\,]]] \\
\mathbf{fwd}(x_{chn}) &\triangleq \,!\,\mathit{tovar}(\texttt{get}\ \mathit{tovar}.\texttt{out.in}\ x_{chn}.\texttt{put}\ \mathit{tovar})[\,] \\
\mathbf{fwd}(l) &\triangleq \,!\,\mathit{toloc}(\texttt{get}\ \mathit{toloc}.\texttt{out.in}\ l.\texttt{dis})[\,] \\
\mathbf{fwd}(x_{loc}) &\triangleq \,!\,\mathit{toloc}(\texttt{get}\ \mathit{toloc}.\texttt{out.in}\ x_{loc}.\texttt{put}\ \mathit{toloc})[\,] \\
\mathbf{Var}(x_{chn}) &\triangleq x_{chn}\langle \mathbf{C_{vn}}\rangle[\,] \\
\mathbf{Var}(x_{loc}) &\triangleq x_{loc}\langle \mathbf{C_{vl}}\rangle[\,] \\
\mathbf{B}(l) &\triangleq \,!\,\mathit{back}(\texttt{put}\ \mathit{tochn})[\mathit{toloc}(\texttt{get}\ \mathit{toloc}.\texttt{get}\ \mathit{toloc}. \\
&\qquad\qquad\qquad\qquad \texttt{sig}\ s.\texttt{out.in}\ l.\texttt{wait}\ s.\texttt{dis})[\,]] \\
\mathbf{C_l} &\triangleq \mathit{toloc}^{\ddagger} \mid \mathit{tonet}^{\uparrow} \mid \mathit{tovar}^{\uparrow} \mid \mathit{comm}^{-} \mid \mathit{cont}^{-} \\
\mathbf{C_n} &\triangleq \mathit{tochn}^{\rotatebox[origin=c]{180}{$\dagger$}} \mid \mathit{comm}^{\uparrow} \\
\mathbf{C_{vl}} &\triangleq \mathit{toloc}^{\updownarrow} \mid \mathit{tovar}^{\rotatebox[origin=c]{180}{$\dagger$}} \\
\mathbf{C_{vn}} &\triangleq \mathit{toloc}^{\uparrow} \mid \mathit{tovar}^{\ddagger}
\end{aligned}
$$

We introduced more system agent names in the encoding. Apart from *comm* and *cont*, which are similar to their counterpart in the encoding of π, we use four different routing agents: *tonet*, *toloc*, *tochn*, and *tovar*. They route cargos respectively to the network level, inside locations, inside channels, or

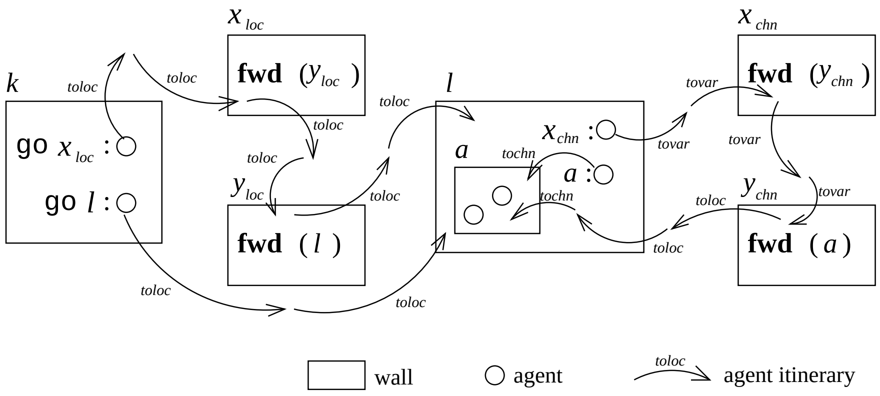

Fig. 6. Wall layouts and agent itineraries in the encoding of Dπ.

inside redirection variables. We always use name s for the signal agent name. The encoding could be better understood with Figure 6 illustrating how the encoding works.

In Figure 6, two locations k and l, together with four redirection ambients are located in the first layer. A channel a is located inside location l. Processes are indicated by circles, and their migration trials are illustrated with arcs. Agent names during migration are shown next to these arcs.

The process $\texttt{go}\ x_{loc}.P$ inside location k reaches location l after it visits redirection ambient $x_{loc}\langle \mathbf{C_{vl}} \rangle[\mathbf{fwd}\ y_{loc}]$ and $y_{loc}\langle \mathbf{C_{vl}} \rangle[\mathbf{fwd}\ l]$, while process $\texttt{go}\ l.P$ can go to location l directly. In location l, a communication process on channel x_{chn} first goes out of l and visit redirection ambient $x_{chn}\langle \mathbf{C_{vn}} \rangle[\mathbf{fwd}\ y_{chn}]$ and $y_{chn}\langle \mathbf{C_{vn}} \rangle[\mathbf{fwd}\ a]$, and then goes back to location l and enters channel ambient a, while the communication process on channel a can move directly into channel ambient a.

The encoding is rather complex due to different kinds of redirection ambients. The use of replicated agents *back* inside each location is essential to bring communication processes back after they visit redirection ambients. The use of "`wait`" and "`sig`" prevents possible interference between different communication processes inside redirection ambients. Moreover, the encoding is not very satisfying if we consider the distributed nature of Dπ locations. The top-level redirection ambients make up a central service point that every location depends on for its computation. This should be avoided since a distributed network like Dπ should have as few centralized control as possible. This problem will be alleviated in the next section with some modification to $\texttt{PR}^-$.

6 Towards a routing calculus

In the Dπ encoding, it is crucial for an agent to remember where it was when visiting redirection ambients. This section extends $\texttt{PR}^-$ and include a primitive called *out-binding* that enables agents to come back easily, so as to simplify

Capabilities:

$$
\begin{array}{lll}
M, N ::= \mathtt{in}\ u.M & \text{move in} \\
\quad\mid\ \mathtt{out}(x).M & \text{move out} \\
\quad\mid\ \mathtt{get}.M & \text{get some cargo} \\
\quad\mid\ \mathtt{put} & \text{put cargo} \\
\quad\mid\ \mathtt{dis} & \text{dissolve}
\end{array}
$$

Values: $u ::= n \mid x$

Processes:

$$
\begin{array}{lll}
P, Q ::= \mathbf{0} & \text{empty} \\
\quad\mid\ P \mid Q & \text{composition} \\
\quad\mid\ !\,P & \text{replication} \\
\quad\mid\ (\nu n)P & \text{restriction} \\
\quad\mid\ M\langle P\rangle & \text{anony. agent} \\
\quad\mid\ n[P] & \text{wall}
\end{array}
$$

Evaluation context:

$$E \quad ::= \quad - \quad \mid \quad E \mid P \quad \mid \quad (\nu n)E \quad \mid \quad n[E]$$

Reduction axioms:

$$
\begin{array}{rcl}
n[P] \mid \mathtt{in}\ n.M\langle Q\rangle & \longrightarrow & n[P \mid M\langle Q\rangle] \\
n[P \mid \mathtt{out}(x).M\langle Q\rangle] & \longrightarrow & n[P] \mid M\{x := n\}\langle Q\{x := n\}\rangle \\
\mathtt{dis}\langle P\rangle & \longrightarrow & P \\
\mathtt{get}.M\langle P\rangle \mid \mathtt{put}\langle Q\rangle & \longrightarrow & M\langle P \mid Q\rangle
\end{array}
$$

Fig. 7. The syntax and reduction semantics of the wagon calculus

the previous sophisticated routing protocols. With out-binding, the primitives `wait` and `sig` added in the encoding of Dπ are not needed anymore.

A second alternation to the $\mathtt{PR}^-$ is the use of anonymous agents. We write $M\langle P\rangle$ instead of $a(M)[P]$, we drop controllers in walls, and we forbid reduction inside cargos. We call the simplified version the *wagon calculus*. It's syntax and reduction semantics are defined in Figure 7.

In process $\mathtt{out}(x).M\langle P\rangle$, variable x in M and P are bound. Substitution is made to both M and P after the agent moves out. We sometime abbreviate $\mathtt{out}(x)$ as `out` if the bind variable x doesn't occur in M and P.

Obviously, wagon is no longer a sub-calculus of ROAM.

Since wagon has also the name substitution primitive. It is easy to write the encoding of π to wagon.

$$
\begin{array}{rclrcl}
\langle\!\langle \mathbf{0} \rangle\!\rangle & \triangleq & \mathbf{0} & \langle\!\langle P \mid Q \rangle\!\rangle & \triangleq & \langle\!\langle P \rangle\!\rangle \mid \langle\!\langle Q \rangle\!\rangle \\
\langle\!\langle\, !\,P \rangle\!\rangle & \triangleq & !\,\langle\!\langle P \rangle\!\rangle & \langle\!\langle (\nu n)P \rangle\!\rangle & \triangleq & (\nu n)(n[\,] \mid \langle\!\langle P \rangle\!\rangle) \\
\langle\!\langle u\langle v\rangle.P \rangle\!\rangle & \triangleq & \multicolumn{4}{l}{\mathtt{in}\ u.\mathtt{get}.\mathtt{out}.\mathtt{in}\ v.\mathtt{dis}\langle \mathtt{out}.\mathtt{dis}.\langle \langle\!\langle P \rangle\!\rangle \rangle\rangle} \\
\langle\!\langle u(x).P \rangle\!\rangle & \triangleq & \multicolumn{4}{l}{\mathtt{in}\ u.\mathtt{put}\langle \mathtt{out}(x).\mathtt{dis}\langle \langle\!\langle P \rangle\!\rangle \rangle\rangle}
\end{array}
$$

In the above encoding, an output process can route an input process into the channel corresponding to the output value, the input process then read the channel name with out-binding, thus substitutes all the occurrence of the variable with the desired value.

The major advantage of using wagon is demonstrated in the encoding of Dπ. Now we don't need redirection ambients and sophisticated routing protocols to get agents back. Passing channel names as well as location names can both be simulated easily using out-binding.

$$\{\!\{\mathbf{0}\}\!\} \triangleq \mathbf{0} \qquad\qquad \{\!\{\mathtt{stop}\}\!\} \triangleq \mathbf{0}$$

$$\begin{array}{rlrl}
\{\!\{N \mid M\}\!\} & \triangleq \{\!\{N\}\!\} \mid \{\!\{M\}\!\} & \{\!\{P \mid Q\}\!\} & \triangleq \{\!\{P\}\!\} \mid \{\!\{Q\}\!\} \\
\{\!\{(\nu l)N\}\!\} & \triangleq (\nu l)(l[\,] \mid \{\!\{N\}\!\}) & \{\!\{\,!\,P\}\!\} & \triangleq !\,\{\!\{P\}\!\} \\
\{\!\{(\nu_l a)N\}\!\} & \triangleq (\nu a)(\mathtt{in}\ l.\mathtt{dis}\langle a[\,]\rangle \mid \{\!\{N\}\!\}) & \{\!\{(\nu l)P\}\!\} & \triangleq (\nu l)(\mathtt{out}.\mathtt{dis}\langle l[\,]\rangle \mid \{\!\{P\}\!\}) \\
\{\!\{l[\![P]\!]\}\!\} & \triangleq \mathtt{in}\ l.\mathtt{dis}\langle \{\!\{P\}\!\}\rangle & \{\!\{(\nu a)P\}\!\} & \triangleq (\nu a)(a[\,] \mid \{\!\{P\}\!\})
\end{array}$$

$$\begin{array}{rcl}
\{\!\{\mathtt{go}\ u_{loc}.P\}\!\} & \triangleq & \mathtt{out}.\mathtt{in}\ u_{loc}.\mathtt{dis}\langle \{\!\{P\}\!\}\rangle \\
\{\!\{u_{chn}\langle v_{loc}\rangle.P\}\!\} & \triangleq & \mathtt{out}(y).\mathtt{in}\ y.\mathtt{in}\ u_{chn}.\mathtt{get}.\mathtt{out}.\mathtt{out}.\mathtt{in}\ v_{loc}.\mathtt{dis} \\
 & & \quad \langle \mathtt{out}.\mathtt{in}\ y.\mathtt{dis}\langle \{\!\{P\}\!\}\rangle\rangle \qquad y \text{ fresh} \\
\{\!\{u_{chn}\langle v_{chn}\rangle.P\}\!\} & \triangleq & \mathtt{in}\ u_{chn}.\mathtt{get}.\mathtt{out}.\mathtt{in}\ v_{chn}.\mathtt{dis}\langle \mathtt{out}.\mathtt{dis}\langle \{\!\{P\}\!\}\rangle\rangle \\
\{\!\{u_{chn}(x_{loc}).P\}\!\} & \triangleq & \mathtt{out}(y).\mathtt{in}\ y.\mathtt{in}\ u_{chn}.\mathtt{put} \\
 & & \quad \langle \mathtt{out}(x_{loc}).\mathtt{in}\ y.\mathtt{dis}\langle \{\!\{P\}\!\}\rangle\rangle \qquad y \text{ fresh} \\
\{\!\{u_{chn}(x_{chn}).P\}\!\} & \triangleq & \mathtt{in}\ u_{chn}.\mathtt{put}\langle \mathtt{out}(x_{chn}).\mathtt{dis}\langle \{\!\{P\}\!\}\rangle\rangle
\end{array}$$

The output processes either route the input processes to locations, when they want to send location values, or to channels inside locations, when they want to send channel values. Note that for exchanging a location, both the input and output process need to return to their original location after communication, so they need to remember where they were by reading the location name with $\mathtt{out}(y).\mathtt{in}\ y$ at the beginning.

Compared with the encoding into $\mathtt{PR}^-$ in the previous section, this encoding is clearly simpler and doesn't suffer the centralized control problem of the former.

7 Conclusion

The first result of this paper is a simplified encoding of π-calculus into pure ambients. We then study an interesting sub-calculus of ROAM called $\mathtt{PR}^-$. $\mathtt{PR}^-$ syntactically defines a subset of well-behaved ROAM processes, and holds most of its expressiveness. We even managed to give an encoding of Dπ in it. Although reference [CDGS03] gives an encoding of Dπ into a version of ambient calculus with anonymous communication, we are not aware of any previous encoding of Dπ into pure ambients.

To simplify the encoding of π and Dπ in $\mathtt{PR}^-$, we extend $\mathtt{PR}^-$ to wagon, where agents are anonymous and can know their current location names. By such modifications, the wagon calculus exhibits a pure packet routing model of computation, which is simple but expressive. We conjecture that most of the expressive power of ambients lies in this computation model. We plan to study the behavior theory of the wagon calculus in the near future.

The encodings in this paper are not formally proved. Those encodings in $\mathtt{PR}^-$ should be easy, since $\mathtt{PR}^-$ processes are simply well-behaved ROAM processes. Proofs for those encodings in wagon would also be easy, after the development of its equational theory similar to that of ROAM.

References

[CDGS03] M. Coppo, M. Dezani-Ciancaglini, E. Giovannetti, and I. Salvo. M3: Mobility types for mobile processes in mobile ambients. In *CATS 2003*, volume 78 of *ENTCS*, 2003.

[CG00] Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000. An extended abstract appeared in *Proceedings of FoSSaCS '98*: 140–155.

[Gua02] Xudong Guan. *Type system and algebrais theory of robust ambients.* PhD thesis, Department of Computer Science and Engineering, Shanghai Jiao Tong Univ., 2002. in Chinese. An extended abstract in English is avaible at: http://www-sop.inria.fr/mimosa/personnel/Xudong.Guan.

[GYY00] X. Guan, Y. Yang, and J. You. Making ambients more robust. In *Proc. ICS2000*, pages 377–384, 2000.

[GYY01] X. Guan, Y. Yang, and J. You. Typing evolving ambients. *Information Processing Letters*, 80(5):265–270, 2001.

[HR98] Matthew Hennessy and James Riely. Resource access control in systems of mobile agents. In Uwe Nestmann and Benjamin C. Pierce, editors, *Proceedings of HLCL '98*, volume 16.3 of *ENTCS*, pages 3–17. Elsevier Science Publishers, 1998. Full version as CogSci Report 2/98, University of Sussex, Brighton.

[LS00] Francesca Levi and Davide Sangiorgi. Controlling interference in ambients. In *Proceedings of POPL '00*, pages 352–364. ACM, January 2000.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I/II. *Journal of Information and Computation*, 100:1–77, September 1992.

[SW01] Davide Sangiorgi and David Walker. *The π-calculus: a Theory of Mobile Processes.* Cambridge University Press, 2001. To appear.

[Zim00] Pascal Zimmer. On the expressiveness of pure mobile ambients. In Luca Aceto and Björn Victor, editors, *Preliminary Proceedings of EXPRESS '00*, volume NS-00-2 of *BRICS Notes Series*, pages 81–104, 2000. The Final Proceedings will be published as volume 39 of ENTCS, Elsevier Science Publishers.

A Previous encodings

As an appendix, we list here two previous encodings of π-calculus in pure ambients.

A.1 Zimmer's encoding in SA [Zim00]

$$
\begin{aligned}
\textbf{server}\ n.P &\triangleq\ !\,\mathit{enter}[\mathtt{in}\ n.\overline{\mathtt{open}}\ \mathit{enter}.P]\\
\textbf{request}\ rw\ n &\triangleq\ \mathtt{in}\ n.\overline{\mathtt{in}}\ rw.\mathtt{open}\ \mathit{enter}\\
\textbf{request}\ rw\ x &\triangleq\ \mathtt{in}\ x.\overline{\mathtt{in}}\ rw.\mathtt{open}\ \mathit{enter}.\mathtt{out}\ x\\
\textbf{fwd}\ \ u &\triangleq\ \textbf{server}\ \mathit{write}.\textbf{request}\ \mathit{write}\ u \mid \textbf{server}\ \mathit{read}.\textbf{request}\ \mathit{read}\ u\\
n\ \textbf{be}\ m.P &\triangleq\ m[\mathtt{out}\ n.\overline{\mathtt{in}}\ m.(\mathtt{open}\ n \mid P)] \mid \overline{\mathtt{out}}\ n.\mathtt{in}\ m.\overline{\mathtt{open}}\ n\\
\textbf{allowIO}\ n &\triangleq\ !\,\overline{\mathtt{in}}\ n \mid !\,\overline{\mathtt{out}}\ n\\
\{\![\mathbf{0}]\!\} &\triangleq\ \mathbf{0}\\
\{\![P \mid Q]\!\} &\triangleq\ \{\![P]\!\} \mid \{\![Q]\!\}\\
\{\![\,!\,P]\!\} &\triangleq\ !\,\{\![P]\!\}\\
\{\![(\nu n)P]\!\} &\triangleq\ (\nu n)\ (\ n[\ \textbf{allowIO}\ n\\
&\qquad\qquad \mid\ \textbf{server}\ \mathit{read}.(\nu p)\\
&\qquad\qquad\qquad (\overline{\mathtt{out}}\ \mathit{read}.\mathit{read}\ \textbf{be}\ p.\overline{\mathtt{in}}\ p.\mathtt{out}\ n.p\ \textbf{be}\ \mathit{read}\\
&\qquad\qquad\qquad \mid\ \mathit{enter}[\mathtt{out}\ \mathit{read}.\mathtt{in}\ \mathit{write}.\overline{\mathtt{open}}\ \mathit{enter}.\\
&\qquad\qquad\qquad\qquad\quad \mathtt{in}\ p.\overline{\mathtt{open}}\ \mathit{write}])]\\
&\qquad\quad \mid \{\![P]\!\})\\
\{\![u\langle v\rangle.P]\!\} &\triangleq\ (\nu p)\ (\ \mathit{write}\ [\ \textbf{request}\ \mathit{write}\ u\\
&\qquad\qquad\qquad \mid \textbf{fwd}\ \ v\\
&\qquad\qquad\qquad \mid p[\mathtt{out}\ \mathit{read}.\overline{\mathtt{open}}\ p.\{\![P]\!\}]]\\
&\qquad\quad \mid \mathtt{open}\ p)\\
\{\![u(x).P]\!\} &\triangleq\ (\nu p)\ (\ \ \mathit{read}\ [\ \textbf{request}\ \mathit{read}\ u\\
&\qquad\qquad\qquad \mid \mathtt{open}\ \mathit{write}.\overline{\mathtt{out}}\ \mathit{read}.(\nu x)\ \mathit{read}\ \textbf{be}\ x.\\
&\qquad\qquad\qquad\qquad (\ \overline{\mathtt{out}}\ x.\textbf{allowIO}\ x\\
&\qquad\qquad\qquad\qquad \mid p[\mathtt{out}\ x.\overline{\mathtt{open}}\ p.\{\![P]\!\}])]\\
&\qquad\quad \mid \mathtt{open}\ p)
\end{aligned}
$$

A.2 Our previous encoding in ROAM [Gua02]

$$
\begin{aligned}
\textbf{server}\ rw.P &\triangleq\ !\,\mathit{enter}[\ \mathtt{in}\ rw.\overline{\mathtt{open}}.P]\\
\textbf{request}\ n &\triangleq\ \mathtt{in}\ n.\overline{\mathtt{in}}\ \mathit{enter}.\mathtt{open}\ \mathit{enter}\\
\textbf{request}\ x &\triangleq\ \mathtt{in}\ x.\overline{\mathtt{in}}\ \mathit{enter}.\mathtt{out}\ x.\mathtt{open}\ \mathit{enter}\\
\textbf{fwd}\ \ M &\triangleq\ \textbf{server}\ r.\textbf{request}\ M \mid \textbf{server}\ w.\textbf{request}\ M\\
\textbf{allowIO} &\triangleq\ !\,\overline{\mathtt{in}}\ r \mid !\,\overline{\mathtt{out}}\ r \mid !\,\overline{\mathtt{in}}\ w \mid !\,\overline{\mathtt{out}}\ w\\
\textbf{wHead} &\triangleq\ \mathtt{in}\ r.\overline{\mathtt{open}}\\
\textbf{wTail}(M,\ P) &\triangleq\ \quad c\ [\ \mathtt{out}\ r.\overline{\mathtt{open}}.\langle\!\langle P\rangle\!\rangle]\\
&\qquad \mid w_1[\ \mathtt{in}\ r_1.\overline{\mathtt{open}}.(\textbf{fwd}\ \ M \mid \textbf{allowIO})\,]\\
\textbf{rHead}(n) &\triangleq\ \overline{\mathtt{in}}\ w.\mathtt{out}\ n.\overline{\mathtt{out}}\ c.\mathtt{open}\ w.\overline{\mathtt{out}}\ c.\mathtt{open}\ r_2.\overline{\mathtt{open}}\\
\textbf{rTail}(x,\ P) &\triangleq\ (\nu x)
\end{aligned}
$$

$$
\begin{array}{rcl}
 & & (\;\; c\;[\;\mathtt{out}\;r.\overline{\mathtt{open}}.\langle\!\langle P\rangle\!\rangle]\\
 & & \mid\;\; r_1\,[\;\overline{\mathtt{in}}\;w_1.\mathtt{in}\;x.\overline{\mathtt{open}}\,]\\
 & & \mid\;\; x\;[\;\overline{\mathtt{in}}\;r_1.\overline{\mathtt{out}}\;r_2.\mathtt{open}\;r_1.\mathtt{open}\;w_1\\
 & & \qquad \mid r_2[\;\mathtt{out}\;x.\overline{\mathtt{open}}\,]\,]\;)\\
\mathbf{chn}(n) & \triangleq & \mathbf{allowIO} \mid \mathbf{server}\;w.\mathbf{wHead} \mid \mathbf{server}\;r.\mathbf{rHead}(n)\\
\langle\!\langle \mathbf{0}\rangle\!\rangle & \triangleq & \mathbf{0}\\
\langle\!\langle P \mid Q\rangle\!\rangle & \triangleq & \langle\!\langle P\rangle\!\rangle \mid \langle\!\langle Q\rangle\!\rangle\\
\langle\!\langle\, !\,P\rangle\!\rangle & \triangleq & !\,\langle\!\langle P\rangle\!\rangle\\
\langle\!\langle(\nu n)P\rangle\!\rangle & \triangleq & (\nu n)(n[\mathbf{chn}(n)] \mid \langle\!\langle P\rangle\!\rangle)\\
\langle\!\langle M\langle M'\rangle.P\rangle\!\rangle & \triangleq & w[\mathbf{request}\;M \mid \mathbf{wTail}(M',\;P)]\\
\langle\!\langle M(x).P\rangle\!\rangle & \triangleq & r[\mathbf{request}\;M \mid \mathbf{rTail}(x,\;P)] \mid \mathtt{open}\;c \mid \mathtt{open}\;c \mid \mathtt{open}\;r
\end{array}
$$

B Comparing the Three Encodings

This appendix gives some comparisons of the three encodings, namely Zimmer00, Guan02, and the one in this paper (Guan03). We use two metrics in our comparison: *size* and *complexity*.

For easy approximation, we define the size of an encoding to be the total number of actions (including macros) used in the right side of "$\triangleq$". This is surely not very accurate. It just serves as a fast approximation.

We define complexity to be the number of reduction steps in the target language needed to simulate one π-reduction. Since a π-reduction in ambient encoding can be divided roughly into the following three stages, we will further list the steps required in each of these stages.

Redirection : I/O processes interacting with redirection ambients (zero or more redirections may be needed to enter the next stage);

Pre-communication : I/O processes entering the right communication channel and getting ready before they select each other;

Post-communication : I/O processes selecting each other and making the new redirection ambients.

As a means to understand the mechanisms of each encoding, we list below the detailed reduction steps in each of these stages. Each reduction step is represented with the subject, the action, and the object (we use the surrounding ambient as the subject of "`open`"). Those reduction steps that can be executed in parallel are listed in parallel. We distinguish actions taken by input and output processes in the first two stages.

Redirection :

Zimmer00 (8 steps):

$$\begin{cases} \mathit{write}\ \mathtt{in}\ x \ \rightarrow\ \mathit{enter}\ \mathtt{in}\ \mathit{write} \ \rightarrow\ \mathit{write}\ \mathtt{open}\ \mathit{enter} \ \rightarrow\ \mathit{write}\ \mathtt{out}\ x \\ \mathit{read}\ \mathtt{in}\ x \ \rightarrow\ \mathit{enter}\ \mathtt{in}\ \mathit{read} \ \rightarrow\ \mathit{read}\ \mathtt{open}\ \mathit{enter} \ \rightarrow\ \mathit{read}\ \mathtt{out}\ x \end{cases}$$

Guan02 (8 steps):

$$\begin{cases} w\ \mathtt{in}\ x \ \rightarrow\ \mathit{enter}\ \mathtt{in}\ w \ \rightarrow\ w\ \mathtt{out}\ x \ \rightarrow\ w\ \mathtt{open}\ \mathit{enter} \\ r\ \mathtt{in}\ x \ \rightarrow\ \mathit{enter}\ \mathtt{in}\ r \ \rightarrow\ r\ \mathtt{out}\ x \ \rightarrow\ r\ \mathtt{open}\ \mathit{enter} \end{cases}$$

Guan03 (8 steps):

$$\begin{cases} \mathit{route}\ \mathtt{in}\ x \ \rightarrow\ \mathit{route}\ \mathtt{in}\ \mathit{route} \ \rightarrow\ \mathit{route}\ \mathtt{open}\ \mathit{route} \ \rightarrow\ \mathit{route}\ \mathtt{out}\ x \\ \mathit{route}\ \mathtt{in}\ x \ \rightarrow\ \mathit{route}\ \mathtt{in}\ \mathit{route} \ \rightarrow\ \mathit{route}\ \mathtt{open}\ \mathit{route} \ \rightarrow\ \mathit{route}\ \mathtt{out}\ x \end{cases}$$

Pre-communication :

Zimmer00 (8 steps):

$$\begin{cases} \mathit{write}\ \mathtt{in}\ n \\ \mathit{read}\ \mathtt{in}\ n \ \rightarrow\ \mathit{enter}\ \mathtt{in}\ \mathit{read} \ \rightarrow\ \mathit{read}\ \mathtt{open}\ \mathit{enter} \ \rightarrow\ (1) \end{cases}$$

$$(1) \ \rightarrow\ \mathit{enter}\ \mathtt{out}\ \mathit{read} \ \rightarrow\ p\ \mathtt{out}\ \mathit{read} \ \rightarrow\ \mathit{read}\ \mathtt{in}\ p \ \rightarrow\ p\ \mathtt{open}\ \mathit{read}$$

Guan02 (6 steps):

$$\begin{cases} w\ \mathtt{in}\ n \ \rightarrow\ \mathit{enter}\ \mathtt{in}\ w \ \rightarrow\ w\ \mathtt{open}\ \mathit{enter} \\ r\ \mathtt{in}\ n \ \rightarrow\ \mathit{enter}\ \mathtt{in}\ r \ \rightarrow\ r\ \mathtt{open}\ \mathit{enter} \end{cases}$$

Guan03 (8 steps):

$$\begin{cases} \mathit{route}\ \mathtt{in}\ n \ \rightarrow\ \mathit{route}\ \mathtt{in}\ \mathit{route} \ \rightarrow\ \mathit{route}\ \mathtt{open}\ \mathit{route} \ \rightarrow\ n\ \mathtt{open}\ \mathit{route} \\ \mathit{route}\ \mathtt{in}\ n \ \rightarrow\ \mathit{route}\ \mathtt{in}\ \mathit{route} \ \rightarrow\ \mathit{route}\ \mathtt{open}\ \mathit{route} \ \rightarrow\ n\ \mathtt{open}\ \mathit{route} \end{cases}$$

Post-communication :

Zimmer00 (15 steps):

$$\mathit{enter}\ \mathtt{in}\ \mathit{write} \ \rightarrow\ \mathit{write}\ \mathtt{open}\ \mathit{enter} \ \rightarrow\ \mathit{write}\ \mathtt{in}\ p \ \rightarrow\ p\ \mathtt{out}\ n \ \rightarrow\ (1)$$

$$(1) \ \rightarrow\ \mathit{read}\ \mathtt{out}\ p \ \rightarrow\ p\ \mathtt{in}\ \mathit{read} \ \rightarrow\ \mathit{read}\ \mathtt{open}\ p \ \rightarrow\ (2)$$

$$(2) \ \rightarrow\ \mathit{read}\ \mathtt{open}\ \mathit{write} \ \rightarrow\ p\ \mathtt{out}\ \mathit{read} \ \rightarrow\ \begin{cases} \mathit{root}\ \mathtt{open}\ p \\ x\ \mathtt{out}\ \mathit{read} \ \rightarrow\ (3) \end{cases}$$

$$(3) \ \rightarrow\ \mathit{read}\ \mathtt{in}\ x \ \rightarrow\ x\ \mathtt{open}\ \mathit{read} \ \rightarrow\ p\ \mathtt{out}\ x \ \rightarrow\ \mathit{root}\ \mathtt{open}\ p$$

Guan02 (14 steps):

$$w\ \mathtt{in}\ r \ \rightarrow\ r\ \mathtt{out}\ n \ \rightarrow\ c\ \mathtt{out}\ r \ \rightarrow\ (1)$$

$$(1) \ \rightarrow\ \begin{cases} \mathit{root}\ \mathtt{open}\ c \\ r\ \mathtt{open}\ w \ \rightarrow\ \begin{cases} c\ \mathtt{out}\ r \ \rightarrow\ \mathit{root}\ \mathtt{open}\ c \\ w_1\ \mathtt{in}\ r_1 \ \rightarrow\ r_1\ \mathtt{in}\ x \ \rightarrow\ r_2\ \mathtt{out}\ x \ \rightarrow\ (2) \end{cases} \end{cases}$$

$$(2) \;\rightarrow\; \begin{cases} r\ \texttt{open}\ r_2 \;\rightarrow\; \mathit{root}\ \texttt{open}\ r \\ x\ \texttt{open}\ r_1 \;\rightarrow\; x\ \texttt{open}\ w_1 \end{cases}$$

Guan03 (9 steps):

$$\mathit{comm}\ \texttt{in}\ \mathit{comm} \;\rightarrow\; \mathit{comm}\ \texttt{open}\ \mathit{comm} \;\rightarrow\; \mathit{comm}\ \texttt{out}\ n \;\rightarrow\; (1)$$

$$(1) \;\rightarrow\; \mathit{comm}\ \texttt{in}\ x \;\rightarrow\; x\ \texttt{open}\ \mathit{comm} \;\rightarrow\; \begin{cases} \mathit{cont}\ \texttt{out}\ x \;\rightarrow\; \mathit{root}\ \texttt{open}\ \mathit{cont} \\ \mathit{cont}\ \texttt{out}\ x \;\rightarrow\; \mathit{root}\ \texttt{open}\ \mathit{cont} \end{cases}$$

We summarized our result in the following table. From this table, we can conclude that the encoding in this paper is simpler than both Zimmer00 and Guan02.

	Zimmer00	Guan02	Guan03
size	47	56	36
complexity*	$8 \times n + 8 + 15$	$8 \times n + 6 + 14$	$8 \times n + 8 + 9$

Note *: We write complexity as $r * n + \mathit{pre} + \mathit{post}$, where r, pre, and post are the reduction steps in the three stages respectively, and n means the average number of redirection steps a process need to perform before reaching its destination channel, which should be a constant in our comparison.