Encoding Channels in Typed Ambients*

Xudong Guan[1] and Jinyuan You[2]

[1] INRIA Sophia Antipolis, France `Xudong.Guan@sophia.inria.fr`
[2] Shanghai Jiao Tong Univ., China `you-jy@cs.sjtu.edu.cn`

Abstract. We start from Zimmer's encoding of the π-calculus to pure ambients [12], and Levi and Sangiorgi's typed equational laws [9] for safe ambients. We presented a typed encoding and reached a stronger operational correspondence proved algebraically with the typed equational laws.

1 Introduction

The ambient calculus [3] is a popular model for mobility and distribution. Ambients can be thought as nodes within a forest, where subtrees can go one level up or down as a whole, and nodes can be dissolved with all the sub-nodes becoming new children of their grand parent. Although local communications [3] and other primitives [1] are introduced for various purposes, the three basic interactions in ambients, namely `in`, `out`, and `open`, form a simple but very expressive theoretical model, as is demonstrated by Zimmer's encoding of π-calculus into pure ambients [12].

To simulate communication and substitution with only ambient movements and opening, the encoding is carried out on an intermediate language called π-*calculus with Explicit Substitutions and Channels* (π_{esc}) that introduces explicit channels as a means of "by-need" redirection services to replace the usually π-substitution after communication. After receiving a value a on channel n, the π_{esc}-process $n(x).P$ will reduce to $(\nu x : a)P$ instead of $P\{a/x\}$, and the explicit substitution structure $(\nu x : a)$ will redirect I/O requests on variable x in process P to a in a lazy manner. While bearing the same expressive power as π, the new calculus is much easier to simulate by ambients.

In the encoding of π_{esc} to ambients, a π_{esc}-channel n is represented by a *channel* ambient with the same name n providing *communication* services to processes. An input (output) process on channel n is represented by an ambient with a fixed name *read* (*write*) requesting services from ambient n. Explicit channel $(\nu x : M)$ (M can be either a channel or another variable) becomes a *variable* ambient x providing *redirection* services to processes. In the encoding, every single π_{esc}-reduction, like explicit substitution, receiving the communication service, or the actual communication, is simulated by a series of ambient movements and opening. In every such series, the first ambient reduction marks

* The first author is supported by EU project 'MIKADO: Mobile Calculi based on Domains', FET-GC IST-2001-32222.

the start of a π_{esc}-reduction, and is called a *principal* reduction. The rests are called *auxiliary* reductions.

Zimmer left open as a conjecture the fact, which he needs for a stronger operational correspondence proof, that auxiliary reductions are confluent. This, in theory, should not be difficult to prove, although this would probably require the assistance of some theorem prover, to manage the huge number of cases encountered in the proof. In this paper, we solve the problem in the following way: we slightly modify Zimmer's encoding, so that we get ambients that are typable in the ETS-MT (*Evolving Type System: Mobility and Threads*) [8] discipline. This is very close to the typing discipline of the Safe Ambients (SA) of Levi and Sangiorgi, so that we can easily adapt the algebraic laws of [9] to our purpose, and then prove that the auxiliary reductions introduced by our encoding are indeed confluent.

1.1 Matching the Encoding with the Equational Laws

In Zimmer's encoding, channel ambients and variable ambients are persistent, with replicated service-carrying ambients named *enter* inside, ready to provide services for the two classes of visiting processes, namely *read* and *write*. A full communication step between *read* and *write* usually consists of the following three phases.

1. **Redirection** (possibly 0 to many times): Ambient *read* (or *write*) enters a variable ambient x, receives redirection service from the service ambient *enter* containing the new destination ambient M, and then leaves for the new destination. This phase is repeated until a channel ambient is reached.
2. **Meeting**: Ambients *read* and *write* both reach some channel ambient n. Ambient *read* receives a service ambient with the same name *enter* from n, opens it, and releases another service ambient *enter*, carrying a shared secret p to *write*. Ambient *write* receives *enter* from *read* and fetches the shared secret p, while ambient *read* renames itself to p. Finally, ambient *write* moves inside p, and then p renames back to *read*.
3. **Communication**: After *write* moves inside *read*, they will come out of the channel and a new variable ambient is created and ready to provide redirection services for the continuation of the input process.

We try to match as much as possible the ambient reductions above to the following two main categories of algebraic laws in [9]. The *single-threaded* (ST) laws ensure that with a few additional constraints, the reduction between any two ST ambients is confluent, where ST means that an ambient will have at most one top-level action at any point of reduction. The *uniform receptive* laws ensure that the reduction between any ST ambient with an immobile server ambient [3] is confluent as long as the service provided by the server is uniform receptive, that is, the service is always available and of the same quality. Actually, we find that most of the ambients in Zimmer's encoding are either single-threaded or

[3] or a server process for the case of $!\,\texttt{open}\, n\,.\,P$.

immobile, and the replicated services are indeed always available and of the same qualities. Below, we list those ambient reductions in the encoding that can be matched with the laws.

1. **ST reductions**: the opening of ambient *enter* inside *read* and *write*, the moving of *enter* from *read* to *write*, the moving of *write* to p, and the opening of *write* inside p;
2. **Uniform receptive reductions**: the entering (leaving) of ambients *read* and *write* into (from) variable ambient x and channel ambient n, and the entering of the replicated ambient *enter* into ambient *read* (*write*).

As for the renaming operations in the encoding, it is shown in [9] that there exists a sound renaming protocol for ST ambients.

The laws of Levi and Sangiorgi are typed in the sense that we have to make the encoding typable in a type environment where channel ambients n and variable ambients x are immobile, and all those ambients *read*, *write*, *enter*, and p are ST. We set the above as our typing objectives.

1.2 The Choice of ETS-MT and ROAM

Two type systems are proposed in [9] to obtain immobile and ST ambients, respectively. Although we believe that the typing objectives above could also be fulfilled with these type systems, we feel that they are somewhat too complex and heavy for reasoning with mobility and threads in pure ambients. In this paper, we choose ETS-MT, a small but powerful type system that tracks mobility and threads in a much simpler flavor with support for subtyping. With the notion of "evolving types", ETS-MT tracks mobility and threads in a very fine degree, supporting examples like opening mobile ambients in immobile ambients. Finally, message exchange type is not obligatory in typing the encoding in pure ambients.

Our work here is based on robust ambients (ROAM) [7]. There seems to be limited reason in the choice of ROAM instead of SA. Our result here works in both of these calculi. The only difference that we learn from our experience is that SA processes are easier to code with, while ROAM processes are easier to verify.

The rest of the paper is organized as follows. The next section gives a brief review of ROAM and ETS-MT. Section 3 rephrases the equational laws of [9] in ROAM, under the ETS-MT flavor. Section 4 presents the typed encoding and proves the operational correspondence result. Finally, Section 5 concludes the paper. In an appendix, we list the grammar and reduction rules of the π_{esc}-calculus, which would be useful for readers not familiar with Zimmer's encoding. Detailed proofs missing in this paper could be found in [6].

2 A Review of ROAM and ETS-MT

ROAM is a variant of SA in that the co-actions have a slightly different semantics: $\overline{\texttt{in}}$ and $\overline{\texttt{out}}$ explicitly specify the name of the ambients that can use them, and $\overline{\texttt{open}}$ no longer needs a parameter. Given a set of names ranged over by n, we recall below the grammar for ROAM processes (P) and capabilities (M), as well as the reduction rules.

$$P ::= \mathbf{0} \;\big|\; (\nu n:T)P \;\big|\; P \mid P \;\big|\; !P \;\big|\; M.P \;\big|\; n[P]$$
$$M ::= \epsilon \;\big|\; \texttt{in}\ n \;\big|\; \texttt{out}\ n \;\big|\; \texttt{open}\ n \;\big|\; \overline{\texttt{in}}\ n \;\big|\; \overline{\texttt{out}}\ n \;\big|\; \overline{\texttt{open}} \;\big|\; M.M$$

$$m[\,\texttt{in}\ n.P_1 \mid P_2] \mid n[\,\overline{\texttt{in}}\ m.Q_1 \mid Q_2] \longrightarrow n[m[P_1 \mid P_2] \mid Q_1 \mid Q_2]$$
$$n[m[\,\texttt{out}\ n.P_1 \mid P_2] \mid \overline{\texttt{out}}\ m.Q_1 \mid Q_2] \longrightarrow m[P_1 \mid P_2] \mid n[Q_1 \mid Q_2]$$
$$\texttt{open}\ n.P \mid n[\,\overline{\texttt{open}}.Q_1 \mid Q_2] \longrightarrow P \mid Q_1 \mid Q_2$$
$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad \frac{P \longrightarrow P'}{(\nu n:T)P \longrightarrow (\nu n:T)P'}$$
$$\frac{P \longrightarrow P'}{n[P] \longrightarrow n[P']} \qquad \frac{P \equiv P' \quad P' \longrightarrow P'' \quad P'' \equiv P'''}{P \longrightarrow P'''}$$

The structural congruence relation $\equiv$ is standard [3, 9, 7].

ETS-MT is a small type system tracking mobility and threads of ROAM processes and ambients. In ETS-MT, both a process and an ambient has type T, being either a simple type $U = Z^Y$ containing information for mobility (Z, mobile "$\curvearrowright$" or immobile "$\veebar$") and threads (Y, no threads "0", ST "1", or multi-threaded (MT) "ω"), or an evolving type $U[T]$ having a *current* type U representing its current behavior, and a *future* type T standing for the behavior after opening. The notion $\perp$ is only used for ease of representing untypable processes. A capability type W is a context, which can make a process type $W(T)$ after filled with some process type T, after some type arithmetics using simple type operators $\cdot$ and $|$ on mobility and threads [8]. A subtyping relation on process types is defined based on $0 \leq 1 \leq \omega$ and $\veebar \leq \curvearrowright$. The grammar of ETS-MT is listed below.

Z	$::=$	$\veebar \mid \curvearrowright$	mobility:	*immobile, mobile*
Y	$::=$	$0 \mid 1 \mid \omega$	threads:	*no, single, multi*
U	$::=$	Z^Y	pre-type	
T	$::=$	$\perp \mid U \mid U[T]$	process type:	*invalid , simple, evolving*
W	$::=$	$-$	capability type:	*hole*
		$U \cdot W$		*prefixing*
		$T \mid W$		*composition*
		$U[W]$		*evolution*

In [8], ETS-MT gives one thread to the $\texttt{open}$ action, with the typing rule $\frac{\Gamma \vdash n:U[T]}{\Gamma \vdash \texttt{open}\ n:\veebar^1 \cdot (T \mid -)}$. In this paper, we refine this rule to $\frac{\Gamma \vdash n:U[T]}{\Gamma \vdash \texttt{open}\ n:\veebar^0 \cdot (T \mid -)}$, according to the insight of [9] that the $\texttt{open}$ action itself can be assigned with no threads: ambient mobility will not interfere with its top-level open action, as long as the open action won't introduce new top-level actions. It is easy to check that with this modification, the subject reduction theorem of the type system still holds. Below is the typing rules of ETS-MT with the above modification.

$$\frac{\Gamma(n) = T}{\Gamma,\, n : T \vdash n : T} \qquad \frac{}{\Gamma \vdash \epsilon : -} \qquad \frac{\Gamma \vdash n : T,\ M \in \{\texttt{in}\ n, \texttt{out}\ n\}}{\Gamma \vdash M : \curvearrowright^{1} \cdot -}$$

$$\frac{\Gamma \vdash n : T,\ M \in \{\overline{\texttt{in}}\ n, \overline{\texttt{out}}\ n\}}{\Gamma \vdash M : \underline{\vee}^{1} \cdot -} \qquad \frac{\Gamma \vdash n : U[T]}{\Gamma \vdash \texttt{open}\ n : \underline{\vee}^{0} \cdot (T \mid -)}$$

$$\frac{}{\Gamma \vdash \overline{\texttt{open}} : \underline{\vee}^{1}[-]} \qquad \frac{\Gamma \vdash M_1 : W_1,\ \Gamma \vdash M_2 : W_2}{\Gamma \vdash M_1 \,.\, M_2 : W_1(W_2)}$$

$$\frac{}{\Gamma \vdash \mathbf{0} : \underline{\vee}^{0}} \qquad \frac{\Gamma \vdash M : W,\ \Gamma \vdash P : T,\ W(T) \neq \bot}{\Gamma \vdash M \,.\, P : W(T)}$$

$$\frac{\Gamma \vdash P : T_1,\ \Gamma \vdash Q : T_2,\ T_1 \mid T_2 \neq \bot}{\Gamma \vdash P \mid Q : T_1 \mid T_2} \qquad \frac{\Gamma \vdash P : T,\ T \mid T \neq \bot}{\Gamma \vdash\ !P : T \mid T}$$

$$\frac{\Gamma,\, n : T_1 \vdash P : T_2}{\Gamma \vdash (\nu n : T_1)P : T_2} \qquad \frac{\Gamma \vdash n : T,\ \Gamma \vdash P : T}{\Gamma \vdash n[P] : \underline{\vee}^{0}} \qquad \frac{\Gamma \vdash P : T_1,\ T_1 \leq T_2}{\Gamma \vdash P : T_2}$$

3 Equational Laws

The notions of mobility and threads in ETS-MT are closely related with the immobile and ST notions in SA. Moreover, grave interferences are controlled by means of disallowing substructure of $\curvearrowright^{\omega}$ in all the type expressions.

Lemma 1. *There will be no grave interference in process P, if the types of all the ambients in P does not have current type of the form $\curvearrowright^{\omega}$.*

Proof. By contradiction. Suppose there will be grave interference. For each of the 10 forms listed in [9], we can show that at least one ambient has current type $\curvearrowright^{\omega}$. □

Still, slight differences exists between the two type disciplines. To make it more accurate, in the rest of the paper, by saying that an ambient is ST we mean that the current type of the ambient is single-threaded. Immobility in [9] also requires that the ambient will never be opened. To avoid confusion, we say an ambient is *stable* when the ambient won't move and can't be opened. Stable ambients will always have types of the form $\underline{\vee}^{Y}$.

Following [9], we write $P \simeq Q$ for barbed congruent processes P and Q having the same behavior inside any context. In typed setting, a law $\Gamma \triangleright P \simeq Q : T$ means P and Q have the same type T and are indistinguishable in all contexts respecting type information Γ and T. Two forms of typed equational laws with contextual constraints are also studied in [9]. Besides the typing requirements for the testing contexts, the testing contexts must be further limited with additional constraints.

– $\Gamma\{L\} \triangleright P \simeq Q : T$: This form gives a stricter constraints on the capabilities that can appear in the testing contexts. For the set of capablities L, all the capabilites (denoted by μ) in it should not appear in those contexts.
– $\Gamma\{rec : L\} \triangleright P \simeq Q : T$: This form ensures the uniform receptiveness of capabilities in L in that each of these capabilities is always available and always activates the same continuation process after being consumed.

Our equational laws in ETS-MT will use almost the same forms. However, as the reduction semantics of ROAM is slightly different from that of SA, we will use $n[\mu]$ instead of a simple capability μ in L [4]. This means a looser constraint on the observing contexts in that capability μ can appear as long as it won't serve as a top level action of n at any time.

3.1 Single-threaded Laws

We use $P \overset{n[\mu]}{\not\Longrightarrow}$ in making the constraints on the top-level ambients inside an ST ambient. We write $P \overset{n[\mu]}{\not\Longrightarrow}$ for process P, ambient n, and capability μ, if it does not hold that $P \longrightarrow^{*}\equiv (\nu\vec{p})(n[\mu\,.\,P_1 \mid P_2] \mid P_3)$ and $(\{n\} \cup fn(\mu)) \cap \{\vec{p}\} = \emptyset$.

Lemma 2. *Suppose n and m are ST in Γ. We have*

1. $\Gamma\{n[\overline{\mathtt{in}}\, m],\ m[\mathtt{in}\, n]\} \triangleright n[\,\overline{\mathtt{in}}\, m\,.\,P_1 \mid P_2] \mid m[\,\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \simeq n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]\,]$;
2. $\Gamma\{\mathtt{open}\, n,\ n[\overline{\mathtt{open}}]\} \triangleright \mathtt{open}\, n\,.\,P_1 \mid n[\,\overline{\mathtt{open}}\,.\,Q_1 \mid Q_2] \simeq P_1 \mid Q_1 \mid Q_2$;
3. $\Gamma \triangleright n[\,\overline{\mathtt{out}}\, m\,.\,P_1 \mid P_2 \mid m[\,\mathtt{out}\, n\,.\,Q_1 \mid Q_2]\,] \simeq n[P_1 \mid P_2] \mid m[Q_1 \mid Q_2]$, *if* $P_2 \overset{m[\mathtt{out}\, n]}{\not\Longrightarrow}$;
4. $\Gamma \triangleright n[\,\mathtt{open}\, m\,.\,P_1 \mid P_2 \mid m[\,\overline{\mathtt{open}}\,.\,Q_1 \mid Q_2]\,] \simeq n[P_1 \mid P_2 \mid Q_1 \mid Q_2]$, *if* $P_2 \overset{m[\overline{\mathtt{open}}]}{\not\Longrightarrow}$.

Proof. The proof is similar to that of [9], since Lemma 1 guarantees the elimination of grave interference in ST ambients. ☐

Some corollary for restricted ambient name instead of context constraints can also be derived.

Corollary 3. *Suppose n, m are ST in Γ. We have*

1a. $\Gamma \triangleright (\nu n)(n[\,\overline{\mathtt{in}}\, m\,.\,P_1 \mid P_2] \mid m[\,\mathtt{in}\, n\,.\,Q_1 \mid Q_2]\,) \simeq (\nu n)(n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]\,]\,)$
1b. $\Gamma \triangleright (\nu m)(n[\,\overline{\mathtt{in}}\, m\,.\,P_1 \mid P_2] \mid m[\,\mathtt{in}\, n\,.\,Q_1 \mid Q_2]\,) \simeq (\nu m)(n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]\,]\,)$
1c. $\Gamma \triangleright h[n[\,\overline{\mathtt{in}}\, m\,.\,P_1 \mid P_2] \mid m[\,\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \mid \mu\,.\,R] \simeq h[n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]\,] \mid \mu\,.\,R]$, *where μ is of the form $\overline{\mathtt{out}}\, k$ or $\mathtt{open}\, k$.*
2a. $\Gamma \triangleright (\nu n)(\mathtt{open}\, n\,.\,P_1 \mid n[\,\overline{\mathtt{open}}\,.\,Q_1 \mid Q_2]\,) \simeq (\nu n)(P_1 \mid Q_1 \mid Q_2)$
2b. $\Gamma \triangleright h[\,\mathtt{open}\, n\,.\,P_1 \mid n[\,\overline{\mathtt{open}}\,.\,Q_1 \mid Q_2] \mid \mu\,.\,R] \simeq h[P_1 \mid Q_1 \mid Q_2 \mid \mu\,.\,R]$, *where μ is of the form $\overline{\mathtt{out}}\, k$ or $\mathtt{open}\, n'$ ($n' \neq n$).*

[4] with the only exception of $\mathtt{open}\, n$.

3.2 Uniform Receptive Laws

Below we investigate in ROAM the uniform receptiveness laws. If we regard the replicated structure as a kind of service, the uniform receptiveness of that service ensures that it is always available, and every reduction partner of that replicated structure will enjoy the same service provided by it. In this circumstance, the reduction between an uniform receptive structure and the other part of the process is confluent.

For every action μ, we can have two kinds of replicated structures: $n[\,!\,\mu\,.\,P \mid Q]$ and $!\,n[\mu\,.\,P \mid Q]$ with the exception of action $\mathtt{open}$, having only the form $!\,\mathtt{open}\ n\,.\,P$. For simplicity, we use $n[\,!\,\mu]$ to represent replicated structure of the form $n[\,!\,\mu\,.\,P \mid Q]$, and $!\,n[\mu]$ for $!\,n[\mu\,.\,P \mid Q]$. Among all the 11 cases, replications like $n[\,!\,\mathtt{in}\ m]$, $n[\,!\,\mathtt{out}\ m]$, $!\,n[\mathtt{out}\ m]$, $!\,n[\overline{\mathtt{out}}\ m]$, and $n[\,!\,\overline{\mathtt{open}}]$ are of little interests.

For replications $!\,n[\mathtt{in}\ m]$, ambient n is mobile and should be ST to control grave interference. As for the uniformity, while disallowing other ambients n having the same top-level action $\mathtt{in}\ m$ to appear in parallel, we allow the existence of parallel process $n[\mathtt{in}\ m\,.\,P \mid Q']$ for replicated structure $n[\mathtt{in}\ m\,.\,P \mid Q]$ with $Q \longrightarrow^* Q'$, since the reduction of Q won't interfere with the entering of n to the service utilizer m. For $!\,n[\overline{\mathtt{in}}\ m]$ and $!\,n[\overline{\mathtt{open}}]$, n should also be ST to guarantee their uniform receptiveness.

The other three cases, namely $n[\,!\,\overline{\mathtt{in}}\ m]$, $n[\,!\,\overline{\mathtt{out}}\ m]$, and $!\,\mathtt{open}\ n$, have already been studied in [9]. Ambient n should be stable, and there should never be processes, for example $\overline{\mathtt{in}}\ m.P'$ with $P \not\equiv P'$ running as the top-level process of $n[\,!\,\overline{\mathtt{in}}\ m\,.\,P \mid Q]$, to bring a different continuation process P' to the service.

Tab. 1 is a list of these six uniform receptiveness structures in ROAM, together with type constraints and uniformity.

Table 1. Six uniform receptiveness structures in ROAM

replications	**type**	**disallowed parallel processes (uniformity)**
$!\,n[\,\mathtt{in}\ m\,.\,P \mid Q]$	$n : ST$	$n[\,\mathtt{in}\ m\,.\,P \mid Q']$ with $Q \not\longrightarrow^* Q'$
$!\,n[\,\overline{\mathtt{in}}\ m\,.\,P \mid Q]$	$n : ST$	$n[\,\overline{\mathtt{in}}\ m\,.\,P \mid Q']$ with $Q \not\longrightarrow^* Q'$
$!\,n[\,\overline{\mathtt{open}}\,.\,P \mid Q]$	$n : ST$	$n[\,\overline{\mathtt{open}}\,.\,P \mid Q']$ with $Q \not\longrightarrow^* Q'$
$n[\,!\,\overline{\mathtt{in}}\ m\,.\,P \mid Q]$	$n : \mathit{Stable}$	$\overline{\mathtt{in}}\ m\,.\,P'$ inside n with $P' \not\equiv P$
$n[\,!\,\overline{\mathtt{out}}\ m\,.\,P \mid Q]$	$n : \mathit{Stable}$	$\overline{\mathtt{out}}\ m\,.\,P'$ inside n with $P' \not\equiv P$
$!\,\mathtt{open}\ n\,.\,P$		$\mathtt{open}\ n\,.\,P'$ with $P' \not\equiv P$

We give the following uniform receptiveness laws. They have the form $\Gamma\{\,!\,n[\mu]\} \triangleright P \simeq Q$, $\Gamma\{n[\,!\,\mu]\} \triangleright P \simeq Q$, or $\Gamma\{\,!\,\overline{\mathtt{open}}\} \triangleright P \simeq Q$, requiring that each of the replicated services inside $\Gamma\{\}$ is uniform receptive, as is defined in Tab. 1.

Lemma 4. *Suppose m is ST in Γ. We have:*

1. $\Gamma\{\,!\,n[\mathtt{in}\ m]\} \triangleright\ !\,n[\,\mathtt{in}\ m\,.\,P_1 \mid P_2] \mid m[\,\overline{\mathtt{in}}\ n\,.\,Q_1 \mid Q_2] \ \simeq\ m[n[P_1 \mid P_2] \mid Q_1 \mid Q_2\,] \mid !\,n[\,\mathtt{in}\ m\,.\,P_1 \mid P_2]$;

2. $\Gamma\{\,!\,n[\overline{\mathtt{in}}\ m]\} \triangleright\ !\,n[\,\overline{\mathtt{in}}\ m\,.\,P_1 \mid P_2] \mid m[\,\mathtt{in}\ n\,.\,Q_1 \mid Q_2] \ \simeq\ n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2\,]] \mid !\,n[\,\overline{\mathtt{in}}\ m\,.\,P_1 \mid P_2]$;
3. $\Gamma\{\,!\,n[\overline{\mathtt{open}}]\} \triangleright \mathtt{open}\ n\,.\,Q \mid !\,n[\,\overline{\mathtt{open}}\,.\,P_1 \mid P_2] \ \simeq\ Q \mid P_1 \mid P_2 \mid !\,n[\,\overline{\mathtt{open}}\,.\,P_1 \mid P_2]$;
4. $\Gamma\{n[\,!\,\overline{\mathtt{in}}\ m]\} \triangleright n[\,!\,\overline{\mathtt{in}}\ m\,.\,P_1 \mid P_2] \mid m[\,\mathtt{in}\ n\,.\,Q_1 \mid Q_2] \ \simeq\ n[\,!\,\overline{\mathtt{in}}\ m\,.\,P_1 \mid P_2 \mid P_1 \mid m[Q_1 \mid Q_2]\,]$;
5. $\Gamma\{n[\,!\,\overline{\mathtt{out}}\ m]\} \triangleright n[\,!\,\overline{\mathtt{out}}\ m\,.\,P_1 \mid P_2 \mid m[\,\mathtt{out}\ n\,.\,Q_1 \mid Q_2] \ \simeq\ n[\,!\,\overline{\mathtt{out}}\ m\,.\,P_1 \mid P_2 \mid P_1] \mid m[Q_1 \mid Q_2]$;
6. $\Gamma\{\,!\,\mathtt{open}\ m\} \triangleright\ !\,\mathtt{open}\ m\,.\,Q \mid m[\,\overline{\mathtt{open}}\,.\,P_1 \mid P_2] \ \simeq\ !\,\mathtt{open}\ m\,.\,Q \mid Q \mid P_1 \mid P_2$.

Proof. The proof of (1)–(3) can be found in [6]. The proof of (4)–(6) is just like in [9].

4 The New Encoding

In this section, we will present the typed encoding of the π_{esc}-calculus to ROAM, and prove the operational correspondence. The encoding of π will be a by-product.

In π_{esc}, a reduction environment called *tree substitution*, denoted by σ, is introduced to unfold the bounding explicit substitutions in processes. A reduction environment is made up of substitution pairs M/x that can map a variable x to a name n after finite numbers of substitutions. To inhibit undesired process behaviors, Zimmer has introduced formal restrictions to limit the set of valid π_{esc}-processes and the corresponding reduction environment. In this paper, we informally say that (σ, P) is a valid π_{esc}-pair, if process P will always have an explicit channel when needed and only one not appearing after any prefix or replication, and meanwhile, all the free variables in P are contained in σ and can be mapped to a name after finite number of substitutions.

4.1 Typed Encoding

Our encoding is based on Zimmer's with a few technical modification. Firstly, we avoid the use of renaming. Although Levi and Sangiorgi proved the soundness of renaming, it requires that the ambient with the new name be mobile and single-threaded. As a result, the renamings of *read* to p and then to x in Zimmer's encoding need modification, since according to our typing objectives, x must be stable. In the new encoding, we use a few system ambients to achieve the same purpose without renaming. Secondly, instead of using a secret name p for the selection of *read* and *write* in channels, we use a more symmetric way of providing the service for *read* as well as that for *write* in the channel ambients. This simplifies the cases we need to consider in our proof. Another trivial modification is the use of a system ambient named c instead of new names p for the continuation processes after communications.

In the new encoding, ambients r and w stand for the reading and writing processes on channels. Ambient *enter* is the service-carrying ambient inside channel or variable ambients. Ambient c contains the continuation process after channel I/O. Instead of using renaming, system ambients r_1, r_2, and w_1 are used as locks to retain single-threadness in the communication steps between the I/O processes.

We will use the following abbreviations in our encoding.

$$
\begin{array}{lcl}
\textbf{server}\ rw\,.\,P & \triangleq & !\,enter[\,\mathtt{in}\ rw\,.\,\overline{\mathtt{open}}\,.\,P] \\
\textbf{request}\ n & \triangleq & \mathtt{in}\ n\,.\,\overline{\mathtt{in}}\ enter\,.\,\mathtt{open}\ enter \\
\textbf{request}\ x & \triangleq & \mathtt{in}\ x\,.\,\overline{\mathtt{in}}\ enter\,.\,\mathtt{out}\ x\,.\,\mathtt{open}\ enter \\
\textbf{fwd}\ \ M & \triangleq & \textbf{server}\ r\,.\,\textbf{request}\ M \mid \textbf{server}\ w\,.\,\textbf{request}\ M \\
\textbf{allowIO} & \triangleq & !\,\overline{\mathtt{in}}\ r \mid !\,\overline{\mathtt{out}}\ r \mid !\,\overline{\mathtt{in}}\ w \mid !\,\overline{\mathtt{out}}\ w \\
 & & \\
\textbf{wHead} & \triangleq & \mathtt{in}\ r\,.\,\overline{\mathtt{open}} \\
\textbf{wTail}(M,\ P) & \triangleq & \quad c\ \ [\,\mathtt{out}\ r\,.\,\overline{\mathtt{open}}\,.\,\{\!\!\{P\}\!\!\}] \\
 & & \mid\ \ w_1\,[\,\mathtt{in}\ r_1\,.\,\overline{\mathtt{open}}\,.\,(\textbf{fwd}\ \ M \mid \textbf{allowIO})\,] \\
\textbf{rHead}(n) & \triangleq & \overline{\mathtt{in}}\ w\,.\,\mathtt{out}\ n\,.\,\overline{\mathtt{out}}\ c\,.\,\mathtt{open}\ w\,.\,\overline{\mathtt{out}}\ c\,.\,\mathtt{open}\ r_2\,.\,\overline{\mathtt{open}} \\
\textbf{rTail}(x,\ P) & \triangleq & (\nu x : \underline{\vee}^{\,\omega}) \\
 & & (\quad c\ \ [\,\mathtt{out}\ r\,.\,\overline{\mathtt{open}}\,.\,\{\!\!\{P\}\!\!\}] \\
 & & \mid\ \ r_1\ [\,\overline{\mathtt{in}}\ w_1\,.\,\mathtt{in}\ x\,.\,\overline{\mathtt{open}}\,] \\
 & & \mid\ \ x\ \ [\,\overline{\mathtt{in}}\ r_1\,.\,\overline{\mathtt{out}}\ r_2\,.\,\mathtt{open}\ r_1\,.\,\mathtt{open}\ w_1 \\
 & & \qquad \mid r_2[\,\mathtt{out}\ x\,.\,\overline{\mathtt{open}}\,]\,]\,) \\
\textbf{chn}(n) & \triangleq & \textbf{allowIO} \mid \textbf{server}\ w\,.\,\textbf{wHead} \mid \textbf{server}\ r\,.\,\textbf{rHead}(n)
\end{array}
$$

When encoding an input process or an output processes of π_{esc} on channel M, we use ROAM processes $r[\textbf{request}\ \ M \mid \textbf{rTail}(x,\ P)\,]$, $w[\textbf{request}\ \ M \mid \textbf{wTail}(M',\ P)\,]$ respectively, serving as clients requesting some kind of service from the ambient M. For explicit substitution $(\nu x : M)$, or an item M/x in the reduction environment, we use $x[\textbf{fwd}\ \ M \mid \textbf{allowIO}]$, so that a request for channel x can be redirected to a request for channel M.

Given a valid π_{esc}-pair $(\sigma,\ P)$, we define the encoding into ROAM $\{\!\!\{\sigma,\ P\}\!\!\}$ recursively as:

$$
\begin{array}{lcl}
\{\!\!\{\mathbf{0}\}\!\!\} & \triangleq & \mathbf{0} \\
\{\!\!\{P \mid Q\}\!\!\} & \triangleq & \{\!\!\{P\}\!\!\} \mid \{\!\!\{Q\}\!\!\} \\
\{\!\!\{\,!\,P\}\!\!\} & \triangleq & !\,\{\!\!\{P\}\!\!\} \\
\{\!\!\{(\nu n)P\}\!\!\} & \triangleq & (\nu n : \underline{\vee}^{\,\omega})\{\!\!\{P\}\!\!\} \\
\{\!\!\{\overline{M}\langle M'\rangle\,.\,P\}\!\!\} & \triangleq & w\,[\,\textbf{request}\ M \mid \textbf{wTail}(M',\ P)\,] \\
\{\!\!\{M(x)\,.\,P\}\!\!\} & \triangleq & r\ \ [\,\textbf{request}\ M \mid \textbf{rTail}(x,\ P)\,] \mid \mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r \\
\{\!\!\{[n : S]\}\!\!\} & \triangleq & n\ [\,\textbf{chn}(n) \mid \{\!\!\{S\}\!\!\}_n] \mid (\mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r)^k \\
 & & \text{(here } k \text{ is the number of input processes in } S) \\
\{\!\!\{(\nu x : M)P\}\!\!\} & \triangleq & (\nu x : \underline{\vee}^{\,\omega})\,(x\,[\,\textbf{fwd}\ \ M \mid \textbf{allowIO}\,] \mid \{\!\!\{P\}\!\!\}) \\
\{\!\!\{\epsilon\}\!\!\}_n & \triangleq & \mathbf{0}
\end{array}
$$

$$
\begin{array}{lcl}
\{\!\!\{ S \mid S' \}\!\!\}_n & \triangleq & \{\!\!\{S\}\!\!\}_n \mid \{\!\!\{S'\}\!\!\}_n \\
\{\!\!\{ \langle M\rangle . P \}\!\!\}_n & \triangleq & w\,[\,\mathbf{wHead} \mid \mathbf{wTail}(M,\, P)\,] \\
\{\!\!\{ (x) . P \}\!\!\}_n & \triangleq & r\,[\,\mathbf{rHead}(n) \mid \mathbf{rTail}(x,\, P)\,] \\
\{\!\!\{ M_1/x_1 \uplus \cdots \uplus M_k/x_k \}\!\!\} & \triangleq & \quad x_1\,[\,\mathbf{fwd}\ \ M_1 \mid \mathbf{allowIO}\,] \\
 & & \mid \ \cdots \\
 & & \mid \ x_k\,[\,\mathbf{fwd}\ \ M_k \mid \mathbf{allowIO}\,] \\
\{\!\!\{ \sigma,\, P \}\!\!\} & \triangleq & \{\!\!\{\sigma\}\!\!\} \mid \{\!\!\{P\}\!\!\}
\end{array}
$$

We will show that the above encoding satisfies our objective in that all the system ambients have single-threaded current type, and those channel and variable ambients are stable.

We define a *system type environment* Γ_{sys} prescribing the following types for all the system ambients.

$$
\begin{array}{lclcl}
\Gamma_{sys} & \triangleq & r & : & \curvearrowright^1 [\underline{\vee}^{\,0}], \qquad w \ : \ \curvearrowright^1 [\underline{\vee}^{\,0}], \\
 & & c & : & \curvearrowright^1 [\underline{\vee}^{\,0}], \qquad r_1 \ : \ \curvearrowright^1 [\underline{\vee}^{\,0}], \\
 & & r_2 & : & \curvearrowright^1 [\underline{\vee}^{\,0}], \qquad w_1 \ : \ \curvearrowright^1 [\underline{\vee}^{\,\omega}], \\
 & & \mathit{enter} & : & \curvearrowright^1 [\curvearrowright^1 [\underline{\vee}^{\,0}]]
\end{array}
$$

Moreover, for valid π_{esc}-pair $(\sigma,\, P)$, we define a function $\mathit{Env}(\sigma,\, P)$, which returns the type environment that contains Γ_{sys} and gives type $\underline{\vee}^{\,\omega}$ to all the free channels and variables.

$$
\mathit{Env}(\sigma,\, P) \quad \triangleq \quad \Gamma_{sys} \cup \{M : \underline{\vee}^{\,\omega} \mid M \in \mathit{dom}(\sigma) \cup \mathit{fn}(P)\}
$$

The following proposition ensures that our encoding is well-typed in the above typing scheme.

Proposition 5. *For any valid π_{esc}-pair $(\sigma,\, P)$, we have $\mathit{Env}(\sigma,\, P) \vdash \{\!\!\{\sigma,\, P\}\!\!\} : \underline{\vee}^{\,0}$.*

Proof. We show $\mathit{Env}(\sigma,\, P) \vdash \{\!\!\{P\}\!\!\} : \underline{\vee}^{\,0}$, by induction on the structure of P. The final result will follow easily.

The result follows by checking the definition of $\{\!\!\{P\}\!\!\}$ with the typing rules. We only list here a few sample derivation steps.

1. **request** , **server** , and **fwd** :

$$
\dfrac{\dfrac{\dfrac{\dfrac{\mathit{Env}(\sigma,\, P) \vdash \mathit{enter} : \curvearrowright^1 [\curvearrowright^1 [\underline{\vee}^{\,0}]]}{\mathit{Env}(\sigma,\, P) \vdash \mathtt{open}\ \mathit{enter} . \mathbf{0} : \curvearrowright^1 [\underline{\vee}^{\,0}]}}{\mathit{Env}(\sigma,\, P) \vdash \mathbf{request}\ n : \curvearrowright^1 [\underline{\vee}^{\,0}] \quad \mathit{Env}(\sigma,\, P) \vdash \mathbf{request}\ x : \curvearrowright^1 [\underline{\vee}^{\,0}]}}{\mathit{Env}(\sigma,\, P) \vdash \mathtt{in}\ rw . \overline{\mathtt{open}} . \mathbf{request}\ M : \curvearrowright^1 [\curvearrowright^1 [\underline{\vee}^{\,0}]]}}{\dfrac{\mathit{Env}(\sigma,\, P) \vdash \mathbf{server}\ rw . \mathbf{request}\ M : \underline{\vee}^{\,0}}{\mathit{Env}(\sigma,\, P) \vdash \mathbf{fwd}\ \ M : \underline{\vee}^{\,0}}}
$$

2. $c[\cdots]$:

$$\dfrac{\dfrac{Env(\sigma,\,P) \vdash \{\!\{P\}\!\} : \underline{\vee}^{\,0}}{Env(\sigma,\,P) \vdash \texttt{out}\ r\,.\,\overline{\texttt{open}}\,.\,\{\!\{P\}\!\} : \curvearrowright^1 [\underline{\vee}^{\,0}]}}{Env(\sigma,\,P) \vdash c[\,\texttt{out}\ r\,.\,\overline{\texttt{open}}\,.\,\{\!\{P\}\!\}\,] : \underline{\vee}^{\,0}}$$

3. $w_1[\cdots]$:

$$\dfrac{\dfrac{Env(\sigma,\,P) \vdash \mathbf{fwd}\ \ M : \underline{\vee}^{\,0} \quad Env(\sigma,\,P) \vdash \mathbf{allowIO} : \underline{\vee}^{\,\omega}}{Env(\sigma,\,P) \vdash \texttt{in}\ r_1\,.\,\overline{\texttt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO}) : \curvearrowright^1 [\underline{\vee}^{\,\omega}]}}{Env(\sigma,\,P) \vdash w_1[\texttt{in}\ r_1\,.\,\overline{\texttt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})] : \underline{\vee}^{\,0}}$$

4. $\{\!\{\overline{M}\langle M'\rangle\,.\,P\}\!\}$:

$$\dfrac{\dfrac{Env(\sigma,\,P) \vdash \mathbf{request}\ M : \curvearrowright^1 [\underline{\vee}^{\,0}] \quad Env(\sigma,\,P) \vdash c[\cdots] \mid w_1[\cdots] : \underline{\vee}^{\,0}}{Env(\sigma,\,P) \vdash \mathbf{request}\ M \mid \mathbf{wTail}(M',\,P) : \curvearrowright^1 [\underline{\vee}^{\,0}]}}{Env(\sigma,\,P) \vdash \{\!\{\overline{M}\langle M'\rangle\,.\,P\}\!\} : \underline{\vee}^{\,0}}$$

□

4.2 Operational Correspondence

Now that the encoding is presented and typable, we establish an operational correspondence between any valid π_{esc}-pairs $(\sigma,\,P)$ and its encoding $\{\!\{\sigma,\,P\}\!\}$. On the one hand, the encoding should simulate all the behavior of the original process, that is $\sigma : P \longrightarrow_{esc} Q \Longrightarrow \{\!\{\sigma,\,P\}\!\} \longrightarrow R$ and $R \approx \{\!\{\sigma,\,Q\}\!\}$ for some equivalence $\approx$ on ROAM processes. On the other hand, any reduction of the encoding should follow one kind of behavior of the original π_{esc}-process, that is, $\{\!\{\sigma,\,P\}\!\} \longrightarrow R \Longrightarrow \sigma : P \longrightarrow_{esc} Q$ and $R \approx \{\!\{\sigma,\,Q\}\!\}$.

However, we can't use simply the relation $\simeq$ unless all the free ambient names are restricted. Otherwise, the validity of the original π_{esc}-process can be easily destroyed by parallelizing additional channel ambients with duplicated names, or by adding variable ambients that map variables to arbitrary names.

Restricting all the free ambients is a way to make the encoding free from such interferences. However, as an undesirable side-effect, the encoding will not be compositional anymore. Another way is by limiting the observation context to the encoding itself. The observer, with the process being observed inside, has to be some part of the encoding of a (larger) π_{esc}-pair.

Definition 6. *ROAM process R is a valid process under π_{esc} translation context (written* $\mathbf{PTC} \vdash R$*), if there exist ROAM context C and a valid π_{esc}-pair $(\sigma,\,P)$ satisfying $\{\!\{\sigma,\,P\}\!\} \longrightarrow^* C(R)$.*

Naturally, if R is a valid process under π_{esc} translation context, it will be typable under some extension of the system typing environment. Moreover, the extension has the form:

$$Env_\rho(R) \quad \triangleq \quad \{n : \underline{\vee}^{\,\omega} \mid n \in fn(R)\backslash\{enter, r, r_1, r_2, w, w_1, c\}\} \cup \Gamma_{sys}$$

Lemma 7. *If* $\mathbf{PTC} \vdash R$*, then* $Env_\rho(R) \vdash R : T$*.*

Proof. By the definition of $\mathbf{PTC} \vdash R$, there exists C, valid π_{esc}-pair (σ, P) satisfying $\{\!\{\sigma, P\}\!\} \longrightarrow^* C(R)$. By Proposition 5 we know $Env(\sigma, P) \vdash \{\!\{\sigma, P\}\!\} : \underline{\vee}^{\,0}$. From $\{\!\{\sigma, P\}\!\} \longrightarrow^* C(R)$ by subject reduction we know $Env(\sigma, P) \vdash C(R) : \underline{\vee}^{\,0}$. So R must be typable under $Env(\sigma, P)$, that is $Env(\sigma, P) \vdash R : T$, for some T. Moreover, we know $Env_\rho(R) \subseteq Env(\sigma, P)$ and $fn(R) \subseteq dom(Env_\rho(R))$, so by the same type derivation steps we can derive $Env_\rho(R) \vdash R : T$. $\square$

By the definition of valid processes under π_{esc} translation context, we can define the following equivalence under π_{esc} translation context. We say that ROAM processes R and S are *barbed congruent under* π_{esc} *translation context* (written $\mathbf{PTC} \triangleright R \simeq S$), if they are both valid processes under π_{esc} translation context, and they have the same behavior if observed within the π_{esc} translation context. The new congruence is less strict that the normal barb congruence, so we can still use the equational laws developed in the previous section in its proof.

The following lemma ensures that the auxiliary reductions, even some of the principal reductions in valid π_{esc} processes, are indeed confluent. It is routine to check that all the following processes are valid under π_{esc} translation context, so we will use this condition implicitly.

Lemma 8. *The following process congruences hold:*

$$
\begin{array}{llcl}
\textit{1a.} & \mathbf{PTC} & \triangleright & n[\mathbf{chn}(n) \mid \{\!\{S\}\!\}_n] \mid r[\,\mathbf{request}\ n \mid \mathbf{rTail}(x,\ P)] \\
 & & \simeq & n[\mathbf{chn}(n) \mid \{\!\{S\}\!\}_n \mid r[\mathbf{rHead}(n) \mid \mathbf{rTail}(x,\ P)]\,] \\
\textit{1b.} & \mathbf{PTC} & \triangleright & n[\mathbf{chn}(n) \mid \{\!\{S\}\!\}_n] \mid w[\,\mathbf{request}\ n \mid \mathbf{wTail}(M,\ P)] \\
 & & \simeq & n[\mathbf{chn}(n) \mid \{\!\{S\}\!\}_n \mid w[\mathbf{wHead} \mid \mathbf{wTail}(M,\ P)]\,]
\end{array}
$$

$$
\begin{array}{llcl}
\textit{2a.} & \mathbf{PTC} & \triangleright & x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid r[\mathbf{request}\ x \mid \mathbf{rTail}(y,\ P)] \\
 & & \simeq & x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid r[\mathbf{request}\ M \mid \mathbf{rTail}(y,\ P)] \\
\textit{2b.} & \mathbf{PTC} & \triangleright & x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid w[\mathbf{request}\ x \mid \mathbf{wTail}(M',\ P)] \\
 & & \simeq & x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid w[\mathbf{request}\ M \mid \mathbf{wTail}(M',\ P)]
\end{array}
$$

$$
\begin{array}{llcl}
\textit{3.} & \mathbf{PTC} & \triangleright & n[\mathbf{chn}(n) \mid \{\!\{S\}\!\}_n \\
 & & & \quad \mid r[\,\mathsf{out}\ n\,.\,\overline{\mathsf{out}}\ c\,.\,\mathsf{open}\ w\,.\,\overline{\mathsf{out}}\ c\,.\,\mathsf{open}\ r_2\,.\,\overline{\mathsf{open}} \\
 & & & \qquad \mid \mathbf{rTail}(x,\ P_2) \\
 & & & \qquad \mid w[\,\overline{\mathsf{open}} \mid \mathbf{wTail}(M,\ P_1)]\,]\,] \\
 & & & \mid \mathsf{open}\ c \mid \mathsf{open}\ c \mid \mathsf{open}\ r \\
 & & \simeq & n[\mathbf{chn}(n) \mid \{\!\{S\}\!\}_n] \\
 & & & \mid \{\!\{P_1\}\!\} \mid (\nu x : \underline{\vee}^{\,\omega})(x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid \{\!\{P_2\}\!\})
\end{array}
$$

Proof. Easy by applying Lemma 2 and Lemma 4, since we know that $enter$, r, w, r_1, r_2, w_1, and c are all ST, and n, x are stable. We show an example of the proof of (1a). We need only to check the additional conditions required to apply Lemma 4.(4), Lemma 4.(1), and Lemma 2.(4).

Lemma 2.(4) is easy since we know $\mathbf{rTail}(x,\ P) \overset{enter[\overline{\mathsf{open}}]}{\not\Longrightarrow}$.

For Lemma 4.(4), we know from the validity of a π_{esc}-pair that ambient n is unique. Also, by analyzing the top-level actions of ambient n, we know that there is no open actions initially (that is, before any reduction happens). Since only opening another ambient can introduce new top-level actions, it is clear that in any of the reductions, there will not be new top-level actions introduced into n. Hence, the uniform receptiveness of the replicated service $n[\,!\,\overline{\texttt{in}}\ r]$ is guaranteed.

For Lemma 4.(1), we need to ensure that $!\,enter[\texttt{in}\ r]$ inside n are indeed uniform receptive. We consider all the possible top-level ambients inside channel ambient n. From the encoding, we know that: (1) Before any reduction, the top-level ambients inside n can only be either the two replicated *enter* ambients, or the ambients r (or w) representing the input (or output) processes. (2) During any reduction involving n, there will be either r and w entering and leaving n, or the ambients *enter* entering r or w that is inside n. There will never be any ambient coming out into n from r or w, and there will never be any ambient other than r and w coming into n. So we can guarantee the uniform receptiveness of $!\,enter[\texttt{in}\ r]$.

We should point out here that in (3), the opening of c and r by the top-level actions $\texttt{open}\ c$ and $\texttt{open}\ r$ do not strictly satisfy Lemma 4.(6). However, we can take them as a special kind of uniform receptiveness with finite number of services, the number of which just equal to the number of clients requesting such services. $\square$

We let $\xrightarrow{com}_{esc}$ be the π_{esc}-reduction derived by the communication rule, and $\xrightarrow{com}$ to represent the mutual selection step of ambients r and w inside a channel, that is, the one that involves $r[\overline{\texttt{in}}\ w\,.\,\cdots]$ and $w[\texttt{in}\ r\,.\,\cdots]$. With the congruences above, we reach the following operational correspondence result for the typed encoding.

Theorem 9. (operational correspondence) *For any valid π_{esc}-pair $(\sigma,\,P)$, we have:*

1. *If $\sigma : P \longrightarrow_{esc} Q$, then there is ROAM process R satisfying $\{\!\{\sigma,\,P\}\!\} \longrightarrow R$ and $\mathbf{PTC} \rhd R \simeq \{\!\{\sigma,\,Q\}\!\}$;*
2. *If $\{\!\{\sigma,\,P\}\!\} \longrightarrow R$, then*
   - *(a) If $\{\!\{\sigma,\,P\}\!\} \xrightarrow{com} R$, then there is π_{esc}-process Q satisfying $\sigma : P \xrightarrow{com}_{esc} Q$ and $\mathbf{PTC} \rhd \{\!\{\sigma,\,Q\}\!\} \simeq R$;*
   - *(b) Otherwise, $\mathbf{PTC} \rhd \{\!\{\sigma,\,P\}\!\} \simeq R$.*

Proof. The first part can be proved by induction of the reduction rules of π_{esc}. The second is by induction on the structure of P, with the help of Lemma 8. $\square$

Due to the relation between π_{esc} and π, the final encoding of π in ROAM can be easily obtained (only for closed π-processes, otherwise add in parallel to the result $n[\mathbf{chn}(n)]$ for every free channel n):

$$
\begin{aligned}
\langle\!\langle \mathbf{0} \rangle\!\rangle &\triangleq \mathbf{0} \\
\langle\!\langle P \mid Q \rangle\!\rangle &\triangleq \langle\!\langle P \rangle\!\rangle \mid \langle\!\langle Q \rangle\!\rangle \\
\langle\!\langle\, !\,P \rangle\!\rangle &\triangleq\ !\,\langle\!\langle P \rangle\!\rangle
\end{aligned}
$$

$$
\begin{array}{lcl}
\langle\!\langle (\nu n)P \rangle\!\rangle & \triangleq & (\nu n : \underline{\vee}^{\omega})(n[\mathbf{chn}(n)] \mid \langle\!\langle P \rangle\!\rangle) \\
\langle\!\langle \overline{M}\langle M' \rangle \,.\, P \rangle\!\rangle & \triangleq & w[\mathbf{request}\ M \mid \mathbf{wTail}(M',\ P)] \\
\langle\!\langle M(x) \,.\, P \rangle\!\rangle & \triangleq & r[\mathbf{request}\ M \mid \mathbf{rTail}(x,\ P)] \mid \mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r
\end{array}
$$

We obtain the final operational correspondence result by defining a special kind of equality between a π-process P with no free variable and a ROAM process R (written $P \bowtie R$), if there is some π_{esc}-process Q such that $P \equiv_{\pi} [\![Q]\!]$ and $\mathbf{PTC} \triangleright \{\!\{\emptyset,\ Q\}\!\} \simeq R$ (where $[\![.]\!]$ is the encoding of π_{esc} to π defined in [12]). It is easy to check that $P \bowtie \langle\!\langle P \rangle\!\rangle$.

Theorem 10. *Suppose $P \bowtie R$.*

1. *If $P \longrightarrow_{\pi} Q$, then there is R' such that $R \longrightarrow^{+} R'$ and $Q \bowtie R'$;*
2. *If $R \longrightarrow R'$, then*
   (a) *If $R \xrightarrow{com} R'$, then there is Q such that $P \longrightarrow_{\pi} Q$ and $Q \bowtie R'$;*
   (b) *Otherwise, $P \bowtie R'$.*

Proof. By combining the result of Theorem 9 with Zimmer's results on the expressive power of π_{esc} and π. □

5 Conclusion and Related Work

Early work by Gordon and Cardelli [5] proved a context lemma for a special context called *harness* in Mobile Ambients (MA), and enumerated in an activity lemma the possible interactions of a process with its enclosing harness. The results have been used to prove the soundness of the firewall-crossing example in MA. Merro and Hennessy have studied labeled bisimulation in a variant of SA enhanced with passwords [10]. At the time when ambient calculus was firstly proposed, pure MA without communication primitives was shown to be Turing-complete [3], and ambients with local communication was known to simulate name-passing channels [3, 9], until Zimmer's encoding of π in pure ambients. Buss and Zavattaro [2] have studied the expressiveness of the fragment of pure MA without movement, restriction, or both, showing that either removing the movement primitives or the restriction retains the Turing-completeness of pure MA.

The result in this paper can be regarded as an application of the equational laws in [9], in the flavor of ETS-MT. From this paper, we can know that the behavior of ambients with co-actions are much easier to predict, and non-trivial results could be derived algebraically. Although the work here is based on ROAM, both ETS-MT and the encoding can be easily migrated to SA.

From the implementation point of view, single-threadness and stability can be utilized to overcome the heavy synchronization costs [4] needed to realize the three basic ambient reductions [11]. This paper shows that even with a small portion of pure ambients which only allows for either stable ambients or single-threaded ambients, the expressive power of π is still retained, while the implementation costs will be largely reduced. In further work we plan to investigate a new version of the Ambient Calculus, where grave interferences are ruled out purely syntactically. While still being quite expressive, its equational theory would be simpler to present and easier to use, and its implementation costs would be largely reduced.

Acknowledgments: The authors are grateful for the proof readings of Gérard Boudol and Pascal Zimmer on the drafts. The authors have benefited a lot from discussions with Francesca Levi and Davide Sangiorgi.

References

[1] T. Amtoft, A. J. Kfoury, and S. M. Pericas-Geertsen. What are polymorphically-typed ambients? In D. Sands, editor, *ESOP 2001, Genova*, volume 2028 of *Lecture Notes in Computer Science*, pages 206–220, Apr. 2001.

[2] N. Busi and G. Zavattaro. On the expressiveness of movement in pure mobile ambients. In *Proc. FWAN'02*, Electronic Notes in Theoretical Computer Science, 2002.

[3] L. Cardelli and A. D. Gordon. Mobile ambients. In M. Nivat, editor, *Proc. FoSSaCS'98*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer Verlag, 1998.

[4] C. Fournet, J.-J. Lévy, and A. Schmitt. An asynchronous, distributed implementation of mobile ambients. In *Proc. IFIP TCS'2000*, pages 348–364, 2000.

[5] A. D. Gordon and L. Cardelli. Equational properties of mobile ambients. In W. Thomas, editor, *Proc. FoSSaCS'99*, volume 1578 of *Lecture Notes in Computer Science*, pages 212–226, 1999.

[6] X. Guan. *Type system and algebraic theory of robust ambients*. PhD thesis, Department of Computer Science and Engineering, Shanghai Jiao Tong Univ., 2002.

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

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

[9] F. Levi and D. Sangiorgi. Controlling interference in ambients. Short version appeared in *Proc. 27th POPL*, ACM Press, 2000.

[10] M. Merro and M. Hennessy. Bisimulation congruences in safe ambients. In *Proc. POPL'02*, pages 71–80. ACM Press, 2002.

[11] D. Sangiorgi and A. Valente. A distributed abstract machine for Safe Ambients. In *Proc. ICALP'01*, volume 2076 of *Lecture Notes in Computer Science*. Springer Verlag, 2001.

[12] P. Zimmer. On the expressiveness of pure mobile ambients. In L. Aceto and B. Victor, editors, *Proc. EXPRESS'00, the 7th International Workshop on Expressiveness in Concurrency*, pages 81–104, 2000.

A The π_{esc}-Calculus

Grammar (extending the π-calculus with explicit channel $[n : S]$ and explicit substitution $(\nu x : M)$, with S stands for I/O processes inside the correct channel):

$$
\begin{array}{rcl}
P & ::= & \mathbf{0} \;\big|\; P \mid Q \;\big|\; !\,P \;\big|\; \overline{M}\langle M' \rangle \,.\, P \;\big|\; M(x)\,.\,P \;\big|\; (\nu n)P \\
 & & \big|\; [n : S] \;\big|\; (\nu x : M)P \\
M & ::= & n \in \mathbf{Name} \;\big|\; x \in \mathbf{Var} \\
S & ::= & \epsilon \;\big|\; S \mid S' \;\big|\; \langle M \rangle \,.\, P \;\big|\; (x)\,.\,P
\end{array}
$$

Reduction rules:

$$
\frac{x\sigma = M}{\sigma : \overline{x}\langle M' \rangle \,.\, P \longrightarrow_{esc} \overline{M}\langle M' \rangle \,.\, P}
\qquad
\frac{x\sigma = M}{\sigma : x(y)\,.\,P \longrightarrow_{esc} M(y)\,.\,P}
$$

$$
\frac{}{\sigma : [n : S] \mid \overline{n}\langle M \rangle \,.\, P \longrightarrow_{esc} [n : S \mid \langle M \rangle \,.\, P]}
$$

$$
\frac{}{\sigma : [n : S] \mid n(x)\,.\,P \longrightarrow_{esc} [n : S \mid (x)\,.\,P]}
$$

$$
\frac{x \neq M}{\sigma : [n : S \mid \langle M \rangle \,.\, P \mid (x)\,.\,Q] \longrightarrow_{esc} [n : S] \mid P \mid (\nu x : M)Q}
$$

$$
\frac{x \notin dom(\sigma) \quad \{M/x\} \uplus \sigma : P \longrightarrow_{esc} P'}{\sigma : (\nu x : M)P \longrightarrow_{esc} (\nu x : M)P'}
\qquad
\frac{P \longrightarrow_{esc} P'}{P \mid Q \longrightarrow_{esc} P' \mid Q}
$$

$$
\frac{P \longrightarrow_{esc} P'}{(\nu n)P \longrightarrow_{esc} (\nu n)P'}
\qquad
\frac{P \equiv_{esc} P' \quad P' \longrightarrow_{esc} P'' \quad P'' \equiv_{esc} P'''}{P \longrightarrow_{esc} P'''}
$$