Typing Evolving Ambients

Xudong Guan Yiling Yang Jinyuan You
Dept. of Comp. Sci. & Eng., Shanghai Jiao Tong Univ., China, 200030
{guan-xd, yang-yl, you-jy}@cs.sjtu.edu.cn

Abstract

This paper presents a type system that deals with the *type evolution problem* of the *ambient calculus*. It breaks the law that the `open` capability must introduce to the opening process the *exact* type information of the opened ambient. It can also serve as a general type architecture supporting type evolution. A typical usage of the type system is illustrated in the *immobile server* example.

Keywords: Theory of computation; Type system; Ambient calculus

1 Introduction

The calculus of *Mobile Ambients* (MA)[3] is a model for *mobile computation* based upon the notion of *ambients*. An ambient may be thought of as a named location where computation happens. It is also the unit of movement, carrying all the processes and subambients within it: it can enter a neighbouring ambient or go out of its parent ambient; it can be opened so that its contents merge into the opening processes. To obtain richer algebraic theories, variants of MA were proposed. *Mobile Safe Ambients* (SA) [9] introduced CCS style co-actions to control the actions of ambients. In SA, an ambient may traverse or open another ambient n only if at least one process inside n agrees. *RObust AMbients* (ROAM) [8] made further restrictions on reduction rules to guarantee the handshaking between actions and co-actions. Co-actions in ROAM can be consumed only by the specified ambients. In this paper, we use *ambient calculus* to denote MA, SA, or ROAM, when the differences among them are not important.

Various type systems for the ambient calculus have been proposed. Topics studied so far include *message exchange type* and *mobility* [4, 5, 10], *security* [6, 7, 2], *threads* and *algebraic theory* [9, 8], etc. In most of these type systems, the following law is uniformly adopted: the `open` capability must introduce to the opening process the *exact* type information of the opened ambient. For instance, in a type system that tracks the *mobility* of ambients, an ambient that opens a *mobile* ambient can not be typed as *immobile*. In fact, this law is too strict. In many cases, the unleashed process behaves differently from what it has done before. We call this the *type evolution problem* of the ambient calculus. This paper aims to devise an *evolving type system* (ETS) that supports type evolution. In addition, we try to make it a general ETS architecture, to which other type information can be easily added.

Our ETS is based on ROAM. It can also be adopted by SA with minor modifications (, but not for MA, since the $\overline{\texttt{open}}$ capability, which is essential in identifying type evolution, does not exist in MA). Moreover, as the type evolution problem is caused by the opening actions only, pure ROAM (that is, the calculus without message exchange) is enough to present our solution. The message exchange constructions of the full calculus only complicates the system.

The rest of this paper is organized as follows. Section 2 reviews the syntax and reduction semantics of pure ROAM. Section 3 presents a basic ETS called `ETS-MT`, which records only two attributes, *mobility* and *threads*. A typical *immobile server* example is given in Section 4 to demonstrate the application of `ETS-MT`. Finally goes the conclusion.

2 ROAM review

ROAM [8] is a variant of SA, with the parameters of co-actions participating in reduction just like those of the actions.

Let $\mathcal{N}$ be a countable set of names ranged over by n, m. The set of processes $\mathcal{P}$ (ranged over by P, Q) and the set of capabilities $\mathcal{M}$ (ranged over by M) [1] are defined below using a BNF-like grammar:

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

Null process ($\mathbf{0}$), restriction with type value $(\nu n : T)$, parallel composition (|), replication (!) and prefixing (.) are standard. $n[P]$ stands for an ambient named n, with process P running inside. Restriction is the only name binder. $fn(P)$ (resp. $fn(M)$) denotes the set of free names in process P (resp. capability W). We often omit the trailing $\mathbf{0}$ in process $M\,.\,\mathbf{0}$.

A capability M can be an empty string (ϵ), a string of capabilities ($M.M$), or a single action. There are six distinct types of actions in ROAM: entering/exiting ambient n ($\mathtt{in}\, n$/$\mathtt{out}\, n$), allowing ambient n to enter/exit($\overline{\mathtt{in}}\, n$/$\overline{\mathtt{out}}\, n$), opening ambient n ($\mathtt{open}\, n$), or allowing to be opened ($\overline{\mathtt{open}}$). It is important that $\overline{\mathtt{open}}$ does not specify who can consume it. Otherwise, a top level ambient can no longer be opened.

The reduction rules of ROAM are:

$$m[\mathtt{in}\, n\,.\,P_1 \mid P_2] \mid n[\overline{\mathtt{in}}\, m\,.\,Q_1 \mid Q_2] \longrightarrow n[m[P_1 \mid P_2] \mid Q_1 \mid Q_2]$$

$$n[m[\mathtt{out}\, n\,.\,P_1 \mid P_2] \mid \overline{\mathtt{out}}\, m\,.\,Q_1 \mid Q_2] \longrightarrow m[P_1 \mid P_2] \mid n[Q_1 \mid Q_2]$$

$$\mathtt{open}\, n\,.\,P \mid n[\overline{\mathtt{open}}\,.\,Q_1 \mid Q_2] \longrightarrow P \mid Q_1 \mid Q_2$$

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad \frac{P \longrightarrow P'}{(\nu n : T)P \longrightarrow (\nu n : T)P'}$$

$$\frac{P \longrightarrow P'}{n[P] \longrightarrow n[P']} \qquad \frac{P \equiv P' \quad P' \longrightarrow P'' \quad P'' \equiv P'''}{P \longrightarrow P'''}$$

In the last rule, the structural congruence relation ($\equiv$) is the least congruence relation that:

- is a commutative monoid for $\mathbf{0}$ and $|$;

- allows to stretch the scope of restrictions, to permute restrictions, and to cancel a restriction followed only by $\mathbf{0}$:

$$n \notin fn(P) \Rightarrow (\nu n : T)(P \mid Q) \equiv P \mid (\nu n : T)Q$$
$$n \neq m \Rightarrow (\nu n : T)m[P] \equiv m[(\nu n : T)P]$$
$$(\nu n : T_n)(\nu m : T_m)P \equiv (\nu m : T_m)(\nu n : T_n)P$$
$$(\nu n : T)\mathbf{0} \equiv \mathbf{0};$$

- satisfies $!P \equiv\ !P \mid P$, $!\mathbf{0} \equiv \mathbf{0}$, $\epsilon\,.\,P \equiv P$, and $(M\,.\,M')\,.\,P \equiv M\,.\,(M'\,.\,P)$.

[1] As long as syntax is concerned, removing ϵ and $M\,.\,M$ from $\mathcal{M}$ will generate the same set $\mathcal{P}$. We keep them only to show that, with the ability to type string of actions, our type system can be extended easily to the full calculus with message exchange.

3 Evolving type system

The main task of ETS is to record the different *type information* before and after type evolution. Many kinds of type information, such as message exchange type, mobility, threads, and security constraints, can be selectively recorded to yield different type systems. To obtain a general ETS architecture, we use an abstract symbol U called *pre-type* to denote the type information recorded. U can be customized to serve different purposes.

As a first attempt, we present `ETS-MT`, a basic ETS recording only *mobility* and *threads*, which are essential to obtain certain properties of behaviour equivalence [9].

3.1 `ETS-MT`: grammar

We first list the grammar of `ETS-MT` below:

Z	::=	$\veebar$	mobility:	*immobile*
		$\curvearrowright$		*mobile*
Y	::=	0	threads:	*none*
		1		*single*
		ω		*multiple*
U	::=	Z^Y	pre-type	
T	::=	$\perp$	process type:	*invalid*
		U		*simple*
		$U[T]$		*evolving*
W	::=	$\$$	capability type:	*hole*
		$U \cdot_t W$		*prefixing*
		$T \mid_t W$		*composition*
		$U[W]$		*evolution*

In `ETS-MT`, the pre-type set $\mathcal{U}$ is defined on two sets: the mobility set $\mathcal{Z} = \{\veebar,\ \curvearrowright\}$ (ranged over by Z) and the threads set $\mathcal{Y} = \{0,\ 1,\ \omega\}$ (ranged over by Y). A process is *mobile* if it may perform at least one mobility action ($\mathtt{in}$ or $\mathtt{out}$), otherwise it is *immobile.* Threads, the number of concurrent capabilities a process may exhibit, can be *none* (e.g. $\mathbf{0}$), *single* (e.g. $\mathtt{out}\, n\,.\,\mathtt{in}\, m\,.\,\overline{\mathtt{out}}\, p$), or *multiple* (e.g. $\mathtt{in}\, n \mid \overline{\mathtt{out}}\, m$ or $!\,\overline{\mathtt{in}}\, m$).

Ambient names and processes have the same type notion called *process type*, denoted by T, $T \in \mathcal{T}$. $\mathcal{T}$ is recursively defined on $\mathcal{U}$. Invalid processes are given type $\perp$. A process that never exhibits $\overline{\mathtt{open}}$ has a *simple type*. Otherwise, it has an *evolving type* $U[T]$ indicating that the process has a simple type U before exercising $\overline{\mathtt{open}}$ and has type T after that. U is called its *current type* and T its *future type*. For instance, process $\mathtt{in}\, n\,.\,\overline{\mathtt{open}}\,.\,!\,\overline{\mathtt{in}}\, m$ is mobile and single-threaded ($\curvearrowright^1$) before $\overline{\mathtt{open}}$. It becomes immobile and multi-threaded ($\underline{\vee}^\omega$) after that. As a result, it has type $\curvearrowright^1 [\underline{\vee}^\omega]$.

To facilitate the typing of processes formulated by prefixing (.) and parallel composition (|), two type operators $\cdot_t$ and $|_t$ are defined.

Firstly, consider the typing of process $M\,.\,P$ with $P : U[T]$ and $M : U'$ (we here assume M is simple, that is, its effect can be described through some pre-type U'). Since M affects only the current type of $M\,.\,P$, $M\,.\,P$ is typed as $(U' \cdot_u U)[T]$, where $\cdot_u$ is some function on pre-types, defined on top of $\cdot_z$ and $\cdot_y$. The whole operation is denoted by $\cdot_t$.

$\cdot_z \;:\; \mathcal{Z} \times \mathcal{Z} \to \mathcal{Z}$

$\cdot_z$	$\underline{\vee}$	$\curvearrowright$
$\underline{\vee}$	$\underline{\vee}$	$\curvearrowright$
$\curvearrowright$	$\curvearrowright$	$\curvearrowright$

$\cdot_y \;:\; \mathcal{Y} \times \mathcal{Y} \to \mathcal{Y}$

$\cdot_y$	0	1	ω
0	0	1	ω
1	1	1	ω
ω	ω	ω	ω

$$\begin{array}{l} \cdot_u \;:\; \mathcal{U} \times \mathcal{U} \to \mathcal{U} \\ \quad Z_1^{Y_1} \cdot_u Z_2^{Y_2} \quad = \quad (Z_1 \cdot_z Z_2)^{(Y_1 \cdot_y Y_2)} \\ \cdot_t \;:\; \mathcal{U} \times \mathcal{T} \to \mathcal{T} \\ \quad \begin{array}{lcl} U \cdot_t \perp & = & \perp \\ U \cdot_t U' & = & U \cdot_u U' \\ U \cdot_t U'[T] & = & (U \cdot_u U')[T] \end{array} \end{array}$$

Secondly, consider the typing of process $P_1 \mid P_2$. There are four different cases to consider. (i) If both of them have simple types, say U_1 and U_2, $P_1 \mid P_2$ is typed as $U_1 \;|_u\; U_2$, where $|_u$ is an operator like $\cdot_u$ (defined on top of $|_z$ and $|_y$ instead). (ii) If one of them, say P_1, has simple type U and the other has evolving type $U'[T]$, U is cast to both U' and T as P_1 may run either before or after the consumption of $\overline{\mathtt{open}}$ in P_2. (iii) If both of them have evolving types, $P_1 \mid P_2$ is difficult to type, since either of the two parallel $\overline{\mathtt{open}}$ actions may be consumed first. We simply let this case to be invalid ($\perp$) [2]. (iv) If one of them is invalid, the result is invalid too.

$|_z \;:\; \mathcal{Z} \times \mathcal{Z} \to \mathcal{Z}$

$\vert_z$	$\underline{\vee}$	$\curvearrowright$
$\underline{\vee}$	$\underline{\vee}$	$\curvearrowright$
$\curvearrowright$	$\curvearrowright$	$\curvearrowright$

$|_y \;:\; \mathcal{Y} \times \mathcal{Y} \to \mathcal{Y}$

$\vert_y$	0	1	ω
0	0	1	ω
1	1	ω	ω
ω	ω	ω	ω

$$\begin{array}{l} |_u \;:\; \mathcal{U} \times \mathcal{U} \to \mathcal{U} \\ \quad Z_1^{Y_1} \;|_u\; Z_2^{Y_2} \quad = \quad (Z_1 \;|_z\; Z_2)^{(Y_1 |_y Y_2)} \\ |_t \;:\; \mathcal{T} \times \mathcal{T} \to \mathcal{T} \\ \quad \begin{array}{lcl} U \;|_t\; U' & = & U \;|_u\; U' \\ U \;|_t\; U'[T] & = & (U \;|_u\; U')[U \;|_t\; T] \\ U'[T] \;|_t\; U & = & (U' \;|_u\; U)[T \;|_t\; U] \\ U_1[T_1] \;|_t\; U_2[T_2] & = & \perp \\ \perp \;|_t\; T & = & \perp \\ T \;|_t\; \perp & = & \perp \end{array} \end{array}$$

Unlike other existing type systems for the ambient calculus, a *capability type* in ETS, W, is a context of process type. W is a type expression containing a hole ($). When this hole is filled with some process type T, W will be evaluated to some other process type T'. This is nature, as each capability M needs a process P to form a new process $M\,.\,P$. $W(T)$ (resp. $W(W')$) denotes the result of replacing the hole in W with T (resp. W'). $W(T) \in \mathcal{T}$ (resp. $W(W') \in \mathcal{W}$).

`ETS-MT` also adopts a certain level of subtyping. The orders of $\mathcal{Z}$, $\mathcal{Y}$, $\mathcal{U}$, and $\mathcal{T}$ are summarized below:

- They are all reflexive and transitive, and
- $\mathcal{Z} :\; \underline{\vee} \leq \curvearrowright \qquad \mathcal{Y} :\; 0 \leq 1 \leq \omega$

  $\mathcal{U}$ and $\mathcal{T}$:

  $Y_1^{Z_1} \leq Y_2^{Z_2} \iff Y_1 \leq Y_2 \wedge Z_1 \leq Z_2$

  $U_1[T_1] \leq U_2[T_2] \iff U_1 \leq U_2 \wedge T_1 \leq T_2$

Process types with different depth of evolution (, that is, with different nesting level of "[ ]",) are uncomparable.

3.2 `ETS-MT`: typing rules

There are three kinds of type judgements in `ETS-MT`: $\Gamma \vdash \Diamond$ means that Γ is a good type environment; $\Gamma \vdash M : W$ means that M can be typed as W under Γ; $\Gamma \vdash G : T$ means that n or P can be typed as T under Γ ($G \in \mathcal{G} = \mathcal{N} \cup \mathcal{P}$). The type environment Γ is a set whose elements have the form $n : T$. The *domain* of Γ is defined as $dom(\Gamma) = \{n \mid n : T \in \Gamma\}$.

The typing rules are divided into three parts.

[2] In fact, this case can also be typed, suppose $P_1 : T_1$, $P_2 : T_2$, let $T_3 = T_1 \;|_t\; T_2$, and let each evolving type be: $T_i = U_{i0}[U_{i1}[...[U_{in_i}]...]]$, $i = 1..3$, then $U_{3k} = \mathtt{MAX}(U_{1i} \;|_u\; U_{2j})_{i+j=k}$, $k = 0..(n_1 + n_2)$. But this kind of non-determinism is rarely used.

Part 1 Typing rules for good environments and ambient names:

$$\frac{-}{\phi \vdash \diamondsuit} \qquad \frac{\Gamma \vdash \diamondsuit,\ n \notin dom(\Gamma)}{\Gamma,\ n : T \vdash \diamondsuit}$$

$$\frac{\Gamma,\ n : T \vdash \diamondsuit}{\Gamma,\ n : T \vdash n : T}$$

Part 2 Typing rules for capabilities:

$$\frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \epsilon : \$} \qquad \frac{\Gamma \vdash n : T,\ M \in \{\mathtt{in}\, n, \mathtt{out}\, n\}}{\Gamma \vdash M : \curvearrowright^1 \cdot_t \$}$$

$$\frac{\Gamma \vdash n : T,\ M \in \{\overline{\mathtt{in}}\, n, \overline{\mathtt{out}}\, n\}}{\Gamma \vdash M : \underline{\vee}^1 \cdot_t \$}$$

$$\frac{\Gamma \vdash n : U[T]}{\Gamma \vdash \mathtt{open}\, n : \underline{\vee}^1 \cdot_t (T \mid_t \$)} \qquad \frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \overline{\mathtt{open}} : \underline{\vee}^1[\$]}$$

$$\frac{\Gamma \vdash M_1 : W_1,\ \Gamma \vdash M_2 : W_2}{\Gamma \vdash M_1 . M_2 : W_1(W_2)}$$

Part 3 Typing rules for processes:

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

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

$$\frac{\Gamma \vdash P : T,\ T \mid_t T \neq \bot}{\Gamma \vdash\ !P : T \mid_t T} \qquad \frac{\Gamma,\ n : T_1 \vdash P : T_2}{\Gamma \vdash (\nu n : T_1)P : T_2}$$

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

With the help of $\cdot_t$, $|_t$, and type context W, these rules are quite concise.

In part 1, the first two rules judge a good environment. The other rule reads the type of a name in a good environment.

The rules in part 2 cast the context-like capability types to the capabilities in $\mathcal{M}$. ϵ is typed as a hole. Simple movement capability (`in`, `out`, $\overline{\mathtt{in}}$, or $\overline{\mathtt{out}}$) is typed as $Z^1 \cdot_t \$$, where Z is either mobile (`in`, `out`) or immobile ($\overline{\mathtt{in}}$, $\overline{\mathtt{out}}$). `open` and $\overline{\mathtt{open}}$ are treated specially, according to their different effects in type evolution. Consider process $\overline{\mathtt{open}} . P$ with $P : T$. It has evolving type $\underline{\vee}^1[T]$, since it has a simple action ($\overline{\mathtt{open}}$, which is immobile and single-threaded) before being opened and has type T after that. In process $\mathtt{open}\, n . P$ with $P : T$, n must be an ambient having some evolving type $U[T']$. After the consumption of $\mathtt{open}\, n$, there will be the parallel composition of P and the process P' come out of n. Since P' has the future type of n, T', $\mathtt{open}\, n . P$ has type $\underline{\vee}^1 \cdot_t (T' \mid_t T)$. The type of capability string is formulated through context substitution, which is still a context.

Typing rules for processes are given in part 3. $\mathbf{0}$ has type $\underline{\vee}^0$, the bottom of simple type. Prefixing is typed through context substitution. $|_t$ is used to type parallel composition and replication. The typing of restriction is standard. Like $\mathbf{0}$, well-typed ambient (ambient name and the inside process have the same type) never issues any command to its outside world ($\underline{\vee}^0$). Finally, process types can be weakened according to the subtyping order.

From these rules, we can easily deduce that if $\Gamma \vdash P : T$, then $T \neq \bot$. Process with multiple $\overline{\mathtt{open}}$ capabilities running in parallel is untypable, e.g. $!\,\overline{\mathtt{open}}$, $\overline{\mathtt{open}} . P \mid \overline{\mathtt{open}} . Q$.

As an example, let $\Gamma = n : U_n[T_n],\ m : U_m$. We can derive $\Gamma \vdash \mathtt{in}\, m . \overline{\mathtt{in}}\, n . \mathtt{open}\, n : \curvearrowright^1 \cdot_t \underline{\vee}^1 \cdot_t \underline{\vee}^1 \cdot_t (T_n \mid_t \$)$. $\mathtt{open}\, m$ is untypable under Γ, because m is a simple typed ambient (m can never be opened). $n[\mathbf{0}]$ is also untypable under Γ, for n is an evolving ambient (n must exhibit $\overline{\mathtt{open}}$ sooner or later, which is impossible for $n[\mathbf{0}]$).

The soundness of `ETS-MT` is ensured by the following subject reduction theorem.

Theorem 1 *If* $\Gamma \vdash P : T$ *and* $P \longrightarrow Q$, *then* $\Gamma \vdash Q : T'$ *and* $T' \leq T$.

The proof is by induction on the depth of the proof of $P \longrightarrow Q$, a routine check similar to those of [8, 5, 10]. See Appendix A for the detailed proof.

4 Immobile server

In this section, we employ a simple example to demonstrate the application of `ETS-MT`.

In this example, a network server is used to collect data brought by agents.

Server $= s[\,!\,\overline{\mathtt{in}}\, a . \mathtt{open}\, a . \mathbf{Collect}]$
Agent $= a[\mathtt{in}\, s . \overline{\mathtt{open}} . \mathbf{Data}]$

We assume process **Collect** and process **Data** are both immobile.

As there are many agents arriving randomly, the server should be available at any time. This requires that the server be immobile. The server, however, must be typed as mobile without the introduction of type evolution, since it has the potential during its execution to open the mobile ambient a.

Server immobility can be easily achieved in `ETS-MT`. Suppose Γ is a good environment satisfying $\Gamma \vdash \mathbf{Data} : \underline{\vee}^{Y_1}$, $\Gamma \vdash \mathbf{Collect} : \underline{\vee}^{Y_2}$, $\Gamma \vdash s : T_s$, and $\Gamma \vdash a : T_a$. T_s and T_a are deduced as follows [3]:

(1) $\Gamma \vdash s : T_s$ given

(2) $\Gamma \vdash \mathtt{in}\, s : \curvearrowright^1 \cdot_t \$$ by (1)

(3) $\Gamma \vdash \diamondsuit$ given

(4) $\Gamma \vdash \overline{\mathtt{open}} : \underline{\vee}^1[\$]$ by (3)

(5) $\Gamma \vdash \mathtt{in}\, s \,.\, \overline{\mathtt{open}} : \curvearrowright^1 \cdot_t \underline{\vee}^1[\$]$ by (2)(4)

(6) $\Gamma \vdash \mathbf{Data} : \underline{\vee}^{Y_1}$ given

(7) $\Gamma \vdash \mathtt{in}\, s \,.\, \overline{\mathtt{open}} \,.\, \mathbf{Data} : \curvearrowright^1 [\underline{\vee}^{Y_1}]$ by (5)(6)

(8) $T_a = \curvearrowright^1 [\underline{\vee}^{Y_1}]$ by (7)

(9) $\Gamma \vdash \overline{\mathtt{in}}\, a : \underline{\vee}^1 \cdot_t \$$ by (8)

(10) $\Gamma \vdash \mathtt{open}\, a : \underline{\vee}^1 \cdot_t (\underline{\vee}^{Y_1} |_t \$)$ by (8)

(11) $\Gamma \vdash \mathbf{Collect} : \underline{\vee}^{Y_2}$ given

(12) $\Gamma \vdash \mathtt{open}\, a \,.\, \mathbf{Collect} : \underline{\vee}^{1 \cdot_y (Y_1 |_y Y_2)}$ by (10)(11)

(13) $\Gamma \vdash \overline{\mathtt{in}}\, a \,.\, \mathtt{open}\, a \,.\, \mathbf{Collect} : \underline{\vee}^{1 \cdot_y (Y_1 |_y Y_2)}$ by (9)(12)

(14) $\Gamma \vdash !\, \overline{\mathtt{in}}\, a \,.\, \mathtt{open}\, a \,.\, \mathbf{Collect} : \underline{\vee}^{\omega}$ by (13)

(15) $T_s = \underline{\vee}^{\omega}$ by (14)

By (15), we know that the server s can be typed as immobile.

In fact, applying both `ETS-MT` and the algebraic results in [9], the encoding of π-calculus into pure ambient calculus [11] can be proved pure algebraically.

5 Discussion

The type evolution problem has been approached in a number of different ways. The early type paper [5] introduced *objective moves* to let objective moving ambients be opened by immobile ambients. In [7], limited type evolution was acquired by requiring only one $\overline{\mathtt{open}}$ capability to be present in the control process of an ambient. A recent paper [1] also introduces a form of type evolution to track *orderly communication*, during which the type of exchanged message can change over time. Additionally, capabilities in their AC^+ calculus are assigned types that are also 'behaviors with a hole within', like the context-like capability type proposed here. Their work can be thought of as the type evolution of message exchange, while this paper focuses on the type evolution of mobility and threads. Moreover, the solution in this paper has a different motivation and tries to provide a general architecture for ETS. Following this ETS architecture, more type information can be stored in the pre-type to yield powerful type systems supporting type evolution.

References

[1] T. Amtoft, A. Kfoury, and S. Pericas-Geertsen. What are polymorphically-typed ambients? To appear in European Symposium On Programming, April 2-6, 2001.

[2] M. Bugliesi, G. Castagna. Secure safe ambients. To appear in Proceeding of POPL'01.

[3] L. Cardelli, A. Gordon. Mobile ambients. In Proc. FoSSaCS'98, LNCS 1378:140–155, Springer-Verlag, 1998.

[4] L. Cardelli, A. Gordon. Types for mobile ambients. In Proc. POPL'99, pages 79-92. ACM Press. 1999.

[5] L. Cardelli, G. Ghelli, A. Gordon. Mobility types for mobile ambients. In Proc. ICALP'99, LNCS 1644:230–239, Spinger-Verlag, 1999.

[6] L. Cardelli, G. Ghelli, A. Gordon. Ambient groups and mobility types. In Proc. TCS2000, LNCS 1872:333–347, Springer-Verlag, 2000.

[7] M. Dezani-Ciancaglini, I. Salvo. Security types for mobile safe ambients. To appear in Proc. ASIAN'00.

[8] X. Guan, Y. Yang, J. You. Making ambients more robust. In Proc. Intl. Conf. on Software: Theory and Practice, pages 377–384, Beijing, China, 2000.

[9] F. Levi, D. Sangiorgi. Controlling interference in ambients. In Proc. POPL'00, pages 352–364, Boston, Massachusetts, 2000.

[10] P. Zimmer. Subtyping and typing algorithms for mobile ambients. In Proc. FoSSaCS'2000, LNCS 1784:375-390, Springer-Verlag, 2000.

[11] P. Zimmer. On the expressiveness of pure mobile ambients. To appear in Proc. EXPRESS'00.

[3] Since subtyping is allowed here, the results starting from (8) are the minimal value possible

Appendix A - Proof of Theorem 1

This appendix give the full proof of the subject reduction theorem.

A.1 Rules and properties

In order to prove Theorem 1, we first list the structural congruence relation, the reduction rules, and the typing rules below, with each rule a distinct name for reference in the rest of this appendix.

Structural congruence:

(Struct Refl)	$P \equiv P$
(Struct Symm)	$Q \equiv P \Rightarrow P \equiv Q$
(Struct Trans)	$P \equiv Q,\ Q \equiv R \Rightarrow P \equiv R$
(Struct Res)	$P \equiv Q \Rightarrow (\nu n : T)P \equiv (\nu n : T)Q$
(Struct Par)	$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$
(Struct Repl)	$P \equiv Q \Rightarrow !\,P \equiv !\,Q$
(Struct Amb)	$P \equiv Q \Rightarrow n[P] \equiv n[Q]$
(Struct Act)	$P \equiv Q \Rightarrow M\,.\,P \equiv M\,.\,Q$
(Struct Par Zero)	$P \mid \mathbf{0} \equiv P$
(Struct Par Comm)	$P \mid Q \equiv Q \mid P$
(Struct Par Assoc)	$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
(Struct Res Zero)	$(\nu n : T)\mathbf{0} \equiv \mathbf{0}$
(Struct Res Res)	$(\nu n : T_n)(\nu m : T_m)P \equiv (\nu m : T_m)(\nu n : T_n)P$
(Struct Res Par)	$n \notin fn(P) \Rightarrow (\nu n : T)(P \mid Q) \equiv P \mid (\nu n : T)Q$
(Struct Res Amb)	$n \neq m \Rightarrow (\nu n : T)m[P] \equiv m[(\nu n : T)P]$
(Struct Repl Par)	$!\,P \equiv P \mid !\,P$
(Struct Repl Zero)	$!\,\mathbf{0} \equiv \mathbf{0}$
(Struct ϵ)	$\epsilon\,.\,P \equiv P$
(Struct $\,.\,$)	$(M_1\,.\,M_2)\,.\,P \equiv M_1\,.\,(M_2\,.\,P)$

Reduction rules:

(Red In) $m[\mathtt{in}\, n\,.\,P_1 \mid P_2] \mid n[\overline{\mathtt{in}}\, m\,.\,Q_1 \mid Q_2] \longrightarrow n[m[P_1 \mid P_2] \mid Q_1 \mid Q_2]$

(Red Out) $n[m[\mathtt{out}\, n\,.\,P_1 \mid P_2] \mid \overline{\mathtt{out}}\, m\,.\,Q_1 \mid Q_2] \longrightarrow m[P_1 \mid P_2] \mid n[Q_1 \mid Q_2]$

(Red Open) $\mathtt{open}\, n\,.\,P \mid n[\overline{\mathtt{open}}\,.\,Q_1 \mid Q_2] \longrightarrow P \mid Q_1 \mid Q_2$

(Red Par) $\dfrac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}$

(Red Res) $\dfrac{P \longrightarrow P'}{(\nu n : T)P \longrightarrow (\nu n : T)P'}$

(Red Amb) $\dfrac{P \longrightarrow P'}{n[P] \longrightarrow n[P']}$

(Red $\equiv$) $\dfrac{P \equiv P' \quad P' \longrightarrow P'' \quad P'' \equiv P'''}{P \longrightarrow P'''}$

Typing rules:

(Env Empty) $\dfrac{-}{\phi \vdash \diamondsuit}$

(Env n) $\dfrac{\Gamma \vdash \diamondsuit,\ n \notin dom(\Gamma)}{\Gamma,\ n : T \vdash \diamondsuit}$

(ET n) $\dfrac{\Gamma,\ n : T \vdash \diamondsuit}{\Gamma,\ n : T \vdash n : T}$

(ET ϵ) $\dfrac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \epsilon : \$}$

(ET Cap $\curvearrowright$) $\dfrac{\Gamma \vdash n : T,\ M \in \{\mathtt{in}\, n, \mathtt{out}\, n\}}{\Gamma \vdash M : \curvearrowright^1 \cdot_t \$}$

(ET Cap $\underline{\vee}$) $\dfrac{\Gamma \vdash n : T,\ M \in \{\overline{\mathtt{in}}\, n, \overline{\mathtt{out}}\, n\}}{\Gamma \vdash M : \underline{\vee}^1 \cdot_t \$}$

(ET $\mathtt{open}$) $\dfrac{\Gamma \vdash n : U[T]}{\Gamma \vdash \mathtt{open}\, n : \underline{\vee}^1 \cdot_t (T \mid_t \$)}$

(ET $\overline{\mathtt{open}}$) $\dfrac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \overline{\mathtt{open}} : \underline{\vee}^1[\$]}$

(ET Path) $\dfrac{\Gamma \vdash M_1 : W_1,\ \Gamma \vdash M_2 : W_2}{\Gamma \vdash M_1\,.\,M_2 : W_1(W_2)}$

(ET Inact) $\dfrac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \mathbf{0} : \underline{\vee}^0}$

(ET Act) $\dfrac{\Gamma \vdash M : W,\ \Gamma \vdash P : T,\ W(T) \neq \bot}{\Gamma \vdash M\,.\,P : W(T)}$

(ET Par) $\dfrac{\Gamma \vdash P : T_1,\ \Gamma \vdash Q : T_2,\ T_1 \mid_t T_2 \neq \bot}{\Gamma \vdash P \mid Q : T_1 \mid_t T_2}$

(ET Repl) $\dfrac{\Gamma \vdash P : T,\ T \mid_t T \neq \bot}{\Gamma \vdash\ !P : T \mid_t T}$

(ET Res) $$\frac{\Gamma,\ n : T_1 \vdash P : T_2}{\Gamma \vdash (\nu n : T_1)P : T_2}$$

(ET Amb) $$\frac{\Gamma \vdash n : T,\ \Gamma \vdash P : T}{\Gamma \vdash n[P] : \underline{\vee}^0}$$

(ET Sub) $$\frac{\Gamma \vdash P : T_1,\ T_1 \leq T_2}{\Gamma \vdash P : T_2}$$

A few properties of type operators $\cdot_t$ and $|_t$ is useful in the proof. These properties can be derived easily from the definitions of the type operators.

Properties 2 *The following properties hold:*

i) (Par Zero Y) $\underline{\vee} \,|_y\, Y = Y$

(Par Zero Z) $0 \,|_z\, Z = Z$

(Par Zero U) $\underline{\vee}^0 \,|_u\, U = U$

(Par Zero T) $\underline{\vee}^0 \,|_t\, T = T$

ii) (Par Symm Y) $Y_1 \,|_y\, Y_2 = Y_2 \,|_y\, Y_1$

(Par Symm Z) $Z_1 \,|_z\, Z_2 = Z_2 \,|_z\, Z_1$

(Par Symm U) $U_1 \,|_u\, U_2 = U_2 \,|_u\, U_1$

(Par Symm T) $T_1 \,|_t\, T_2 = T_2 \,|_t\, T_1$

iii) (Par Assoc Y) $(Y_1 \,|_y\, Y_2) \,|_y\, Y_3 = Y_1 \,|_y\, (Y_2 \,|_y\, Y_3)$

(Par Assoc Z) $(Z_1 \,|_z\, Z_2) \,|_z\, Z_3 = Z_1 \,|_z\, (Z_2 \,|_z\, Z_3)$

(Par Assoc U) $(U_1 \,|_u\, U_2) \,|_u\, U_3 = U_1 \,|_u\, (U_2 \,|_u\, U_3)$

(Par Assoc T) $(T_1 \,|_t\, T_2) \,|_t\, T_3 = T_1 \,|_t\, (T_2 \,|_t\, T_3)$

iv) (Par Strict Y) $Y_1 \leq Y_2 \Rightarrow Y_1 \,|_y\, Y_3 \leq Y_2 \,|_y\, Y_3$

(Par Strict Z) $Z_1 \leq Z_2 \Rightarrow Z_1 \,|_z\, Z_3 \leq Z_2 \,|_z\, Z_3$

(Par Strict U) $U_1 \leq U_2 \Rightarrow U_1 \,|_u\, U_3 \leq U_2 \,|_u\, U_3$

(Par Strict T) $T_1 \leq T_2 \Rightarrow T_1 \,|_t\, T_3 \leq T_2 \,|_t\, T_3$

Properties 3 *The following properties hold:*

i) (Dot Zero Y) $\underline{\vee} \cdot_y Y = Y$

(Dot Zero Z) $0 \cdot_z Z = Z$

(Dot Zero U) $\underline{\vee}^0 \cdot_u U = U$

(Dot Zero T) $\underline{\vee}^0 \cdot_t T = T$

ii) (Dot Symm Y) $Y_1 \cdot_y Y_2 = Y_2 \cdot_y Y_1$

(Dot Symm Z) $Z_1 \cdot_z Z_2 = Z_2 \cdot_z Z_1$

(Dot Symm U) $U_1 \cdot_u U_2 = U_2 \cdot_u U_1$

iii) (Dot Assoc Y) $(Y_1 \cdot_y Y_2) \cdot_y Y_3 = Y_1 \cdot_y (Y_2 \cdot_y Y_3)$

(Dot Assoc Z) $(Z_1 \cdot_z Z_2) \cdot_z Z_3 = Z_1 \cdot_z (Z_2 \cdot_z Z_3)$

(Dot Assoc U) $(U_1 \cdot_u U_2) \cdot_u U_3 = U_1 \cdot_u (U_2 \cdot_u U_3)$

(Dot Assoc T) $(U_1 \cdot_t U_2) \cdot_t T = U_1 \cdot_t (U_2 \cdot_t T)$

iv) (Dot Strict Y) $Y_1 \leq Y_2 \Rightarrow Y_1 \cdot_y Y_3 \leq Y_2 \cdot_y Y_3$

(Dot Strict Z) $Z_1 \leq Z_2 \Rightarrow Z_1 \cdot_z Z_3 \leq Z_2 \cdot_z Z_3$

(Dot Strict U) $U_1 \leq U_2 \Rightarrow U_1 \cdot_u U_3 \leq U_2 \cdot_u U_3$

(Dot Strict T) $U_1 \leq U_2 \Rightarrow U_1 \cdot_t T \leq U_2 \cdot_t T$

A.2 Lemmas

First, we prove that process type is preserved by structural congruence (Lemma 9), based on a few technical lemmas and the properties above.

Since the subtyping order is reflexive and transitive, we will always consider in the following proofs that there was exactly one subsumption rule (ET Sub) just after each other typing rule, when appropriate (in Lemma 9 (Subject conguence) and Theorem 1 (Subject reduction)).

A formalization of the validity assertion of the typing rules in Section 3 is needed.

Proposition 4 (Validity) *If* $\Gamma \vdash P : T$, *then* $T \neq \bot$.

Proof. By an easy induction of the derivation of $\Gamma \vdash P : T$. ∎

Let $\Gamma \vdash J$ denote any of the three judgements: $\Gamma \vdash \diamondsuit$, $\Gamma \vdash M : W$, or $\Gamma \vdash P : T$. We extend naturally $fn()$ to J.

Lemma 5 *If* $\Gamma,\ n : T \vdash J$, *then* $n \notin dom(\Gamma)$.

Proof. The only way to introduce new elements in the environment is by using the rule (Env n). Since the premise of (Env n) guarantees that the ambient name of the introduced element is distinct, the result follows. ∎

Lemma 6 (Implied judgement) *If* $\Gamma_1,\ \Gamma_2 \vdash J$, *then* $\Gamma_1 \vdash \diamondsuit$.

Proof. By induction on Γ_1.

- If $\Gamma_1 = \phi$, then by (Env Empty), $\phi \vdash \diamondsuit$;
- If $\Gamma_1 = \Gamma_1',\ n : T$, then $\Gamma_1',\ n : T,\ \Gamma_2 \vdash J$. By induction hypothesis, we have $\Gamma_1' \vdash \diamondsuit$. By Lemma 5, we have $n \notin dom(\Gamma_1')$. By (Env n), we can derive $\Gamma_1',\ n : T \vdash \diamondsuit$, that is, $\Gamma_1 \vdash \diamondsuit$.

∎

Lemma 7 (Strengthening) *If* $\Gamma,\ n:T \vdash J$ *and* $n \notin fn(J)$*, then* $\Gamma \vdash J$*.*

Proof. Since $n \notin fn(J)$, we can eliminate all the occurrence of n in J by renaming all the bound names. By this means, $n : T$ will not be used in the inference of $\Gamma,\ n : T \vdash J$. By the same depth of inference, we can derive $\Gamma \vdash J$. ∎

Lemma 8 (Weakening) *If* $\Gamma \vdash J$ *and* $n \notin dom(\Gamma)$*, then* $E,\ n:T \vdash J$*.*

Proof. First, by $\Gamma \vdash J$ and Lemma 6, we have $\Gamma \vdash \diamondsuit$. By (Env n), we have $\Gamma,\ n:T \vdash \diamondsuit$. Based on this, $\Gamma,\ n:T \vdash J$ could be derived by using the same depth of inference as the derivation of $\Gamma \vdash J$. ∎

Lemma 9 (Subject congurence) *If* $\Gamma \vdash P:T$ *and* $P \equiv Q$*, then* $\Gamma \vdash Q:T$*.*

Proof. We prove the following two parts by mutural induction on the derivation of $P \equiv Q$ and $Q \equiv P$. We will not detail part (2) when the reason is the same, i.e. for symmetrical rules.

(1) If $\Gamma \vdash P:T$ and $P \equiv Q$, then $\Gamma \vdash Q:T$;

(2) If $\Gamma \vdash Q:T$ and $P \equiv Q$, then $\Gamma \vdash P:T$.

(Struct Refl) Trivial.

(Struct Symm) For part (1), we have $Q \equiv P$, the result follows by induction hypothesis (2). The proof of part (2) is symmetrical.

(Struct Trans) For part (1), we have $P \equiv R$ and $R \equiv Q$. Suppose $\Gamma \vdash P:T$. By induction hypothesis (1), we have $\Gamma \vdash R:T$, and then $\Gamma \vdash Q:T$. The proof of part (2) is symmetrical.

(Struct Res) For part (1), we have $P = (\nu n:T_n)P_1$, $Q = (\nu n:T_n)Q_1$, and $P_1 \equiv Q_1$. Suppose $\Gamma \vdash (\nu n : T_n)P_1 : T$. This could only be derived by using (ET Sub) and (ET Res), with $\Gamma,\ n:T_n \vdash P_1:T'$ and $T' \leq T$. By $P_1 \equiv Q_1$ and induction hypothesis (1), we have $\Gamma,\ n:T_n \vdash Q_1:T'$. By (ET Res) and (ET Sub), we can derive $\Gamma \vdash (\nu n:T_n)Q_1:T$. The proof of part (2) is symmetrical.

(Struct Par) For part (1), we have $P = P_1 \mid R$, $Q = Q_1 \mid R$, and $P_1 \equiv Q_1$. Suppose $\Gamma \vdash P_1 \mid R:T$. This could only be derived by using (ET Sub) and (ET Par), with $\Gamma \vdash P_1:T_1$, $\Gamma \vdash R:T_2$, and $T_1 \mid_t T_2 \leq T$. By $P_1 \equiv Q_1$ and induction hypothesis (1), we have $\Gamma \vdash Q_1:T_1$. By (ET Par) and (ET Sub), we get $\Gamma \vdash Q_1 \mid R:T$. The proof of part (2) is symmetrical.

(Struct Repl) For part (1), we have $P = !P_1$, $Q = !Q_1$, and $P_1 \equiv Q_1$. Suppose $\Gamma \vdash !P_1:T$. This could only be derived by using (ET Sub) and (ET Repl), with $\Gamma \vdash P_1:T_1$ and $T_1 \mid_t T_1 \leq T$. By $P_1 \equiv Q_1$ and induction hypothesis (1), we have $\Gamma \vdash Q_1:T_1$. By (ET Repl) and (ET Sub), we can derive $\Gamma \vdash !Q_1:T$. The proof of part (2) is symmetrical.

(Struct Amb) For part (1), we have $P = n[P_1]$, $Q = n[Q_1]$, and $P_1 \equiv Q_1$. Suppose $\Gamma \vdash n[P_1]:T$. This could only be derived by using (ET Sub) and (ET Amb), with $\Gamma \vdash n:T_1$, $\Gamma \vdash P_1:T_1$, and $\underline{\vee}^0 \leq T$. By $P_1 \equiv Q_1$ and induction hypothesis (1), we have $\Gamma \vdash Q_1:T_1$. By (ET Amb) and (ET Sub), we get $\Gamma \vdash n[Q_1]:T$. The proof of part (2) is symmetrical.

(Struct Act) For part (1), we have $P = M\,.\,P_1$, $Q = M\,.\,Q_1$, and $P_1 \equiv Q_1$. Suppose $\Gamma \vdash M\,.\,P_1 : T$. This could only be derived by using (ET Sub) and (ET Act), with $\Gamma \vdash M : W$, $\Gamma \vdash P_1 : T_1$, and $W(T_1) \leq T$. By $P_1 \equiv Q_1$ and induction hypothesis (1), we have $\Gamma \vdash Q_1 : T_1$. By (ET Act) and (ET Sub), we can derive $\Gamma \vdash M\,.\,Q_1 : T$. The proof of part (2) is symmetrical.

(Struct Par Zero) Then, we have $P = Q \mid \mathbf{0}$. For part (1), suppose $\Gamma \vdash Q \mid \mathbf{0} : T$. This could only be derived by using (ET Sub) and (ET Par), with $\Gamma \vdash Q : T_1$, $\Gamma \vdash \mathbf{0} : \underline{\vee}^0$, and $T_1 \mid_t \underline{\vee}^0 \leq T$. By (Par Zero T) in Property 2, we have $T_1 \leq T$. Finally, $\Gamma \vdash Q : T$ by (ET Sub). For part (2), suppose $\Gamma \vdash Q : T$. By Lemma 6, we have $\Gamma \vdash \diamondsuit$. By (ET Inact), $\Gamma \vdash \mathbf{0} : \underline{\vee}^0$. Finally, by (ET Par), we can derive $\Gamma \vdash Q \mid \mathbf{0} : T \mid_t \underline{\vee}^0$, that is, $\Gamma \vdash Q \mid \mathbf{0} : T$.

(Struct Par Comm) Then, we have $P = P_1 \mid Q_1$ and $Q = Q_1 \mid P_1$. For part (1), suppose $\Gamma \vdash P_1 \mid Q_1 : T$. This could only be derived by using (ET Sub) and (ET Par), with $\Gamma \vdash P_1 : T_1$, $\Gamma \vdash Q_1 : T_2$, and $T_1 \mid_t T_2 \leq T$. By (ET Par), we have $\Gamma \vdash Q_1 \mid P_1 : T_2 \mid_t T_1$. From (Par Symm T) in Property 2, we know that $T_2 \mid_t T_1 = T_1 \mid_t T_2$. By (ET Sub), we get $\Gamma \vdash Q_1 \mid P_1 : T$. The proof of part (2) is symmetrical.

(Struct Par Assoc) Then, we have $P = (P_1 \mid Q_1) \mid R$ and $Q = P_1 \mid (Q_1 \mid R)$. For part (1), suppose $\Gamma \vdash (P_1 \mid Q_1) \mid R : T$. This could only be derived by using (ET Sub) and (ET Par), with $\Gamma \vdash P_1 \mid Q_1 : T_1$, $\Gamma \vdash R : T_2$, and $T_1 \mid_t T_2 \leq T$. The former judgement could only be derived by using (ET Sub) and (ET Par), with $\Gamma \vdash P_1 : T_3$, $\Gamma \vdash Q_1 : T_4$, and $T_3 \mid_t T_4 \leq T_1$. By (ET Par) twice, we have $\Gamma \vdash P_1 \mid (Q_1 \mid R) : T_3 \mid_t (T_4 \mid_t T_2)$. By (Par Assoc T) in Property 2, we know that $T_3 \mid_t (T_4 \mid_t T_2) = (T_3 \mid_t T_4) \mid_t T_2$. Finally, $\Gamma \vdash P_1 \mid (Q_1 \mid R) : T$ by (ET Sub). The proof of part (2) is symmetrical.

(Struct Res Zero) Then, we have $P = (\nu n : T_n)\mathbf{0}$ and $Q = \mathbf{0}$. For part (1), suppose $\Gamma \vdash (\nu n : T_n)\mathbf{0} : T$. This could only be derived by using (ET Sub) and (ET Res), with $\Gamma,\ n : T_n \vdash \mathbf{0} : T_1$ and $T_1 \leq T$. Since $n \notin fn(\mathbf{0})$, by Lemma 7, we have $\Gamma \vdash \mathbf{0} : T_1$. By (ET Sub), we can derive $\Gamma \vdash \mathbf{0} : T$. For part (2), suppose $\Gamma \vdash \mathbf{0} : T$. By renaming bound names, we can assure $n \notin dom(\Gamma)$. By Lemma 8, we have $\Gamma,\ n : T_n \vdash \mathbf{0} : T$. By (ET Res), we get $\Gamma \vdash (\nu n : T_n)\mathbf{0} : T$.

(Struct Res Res) Then, we have $P = (\nu n : T_n)(\nu m : T_m)R$ and $Q = (\nu m : T_m)(\nu n : T_n)R$. For part (1), suppose $\Gamma \vdash (\nu n : T_n)(\nu m : T_m)R : T$. This could only be derived by using (ET Sub) and (ET Res), with $\Gamma,\ n : T_n \vdash (\nu m : T_m)R : T_1$ and $T_1 \leq T$. For the same reason, it must be true that $\Gamma,\ n : T_n,\ m : T_m \vdash R : T_2$ and $T_2 \leq T_1$. Since type environment is a set with no order. The result follows by applying (ET Res) and (ET Sub) twice each. The proof of part (2) is symmetrical.

(Struct Res Par) Then, we have $P = (\nu n : T_n)(P_1 \mid Q_1)$, $Q = P_1 \mid (\nu n : T_n)Q_1$, and $n \notin fn(P_1)$. For part (1), suppose $\Gamma \vdash (\nu n : T_n)(P_1 \mid Q_1) : T$. This could only be derived by using (ET Sub) and (ET Res), with $\Gamma,\ n : T_n \vdash P_1 \mid Q_1 : T_1$ and $T_1 \leq T$. The former judgement could only be derived by using (ET Sub) and (ET Par), with $\Gamma,\ n : T_n \vdash P_1 : T_2$, $\Gamma,\ n : T_n \vdash Q_1 : T_3$, and $T_2 \mid_t T_3 \leq T_1$. Then, $\Gamma \vdash (\nu n : T_n)Q_1 : T_3$ by (ET Res). Since $n \notin fn(P)$, by Lemma 7, we have $\Gamma \vdash P_1 : T_2$. By (ET Par) and (ET Sub), we get $\Gamma \vdash P_1 \mid (\nu n : T_n)Q_1 : T$. Part (2) follows by a similar analysis, but uses Lemma 8 instead.

(Struct Res Amb) Then, we have $P = (\nu n : T_n)m[R]$, $Q = m[(\nu n : T_n)R]$, and $n \neq m$. For part (1), suppose $\Gamma \vdash (\nu n : T_n)m[R] : T$. This could only be derived by using (ET Sub) and (ET Res), with $\Gamma,\ n : T_n \vdash m[R] : T_1$ and $T_1 \leq T$. The former judgement could only be derived by using (ET Sub) and (ET Amb), with $\Gamma,\ n : T_n \vdash m : T_m$, $\Gamma,\ n : T_n \vdash R : T_m$, and $\underline{\vee}^0 \leq T_1$. Then $\Gamma \vdash (\nu n : T_n)R : T_m$ by (ET Res). Since $n \neq m$, by Lemma 7, we have $\Gamma \vdash m : T_m$. By (ET Amb) and (ET Sub), we get $\Gamma \vdash m[(\nu n : T_n)R] : T$. Part (2) follows by a similar analysis, but uses Lemma 8 instead.

(Struct Repl Par) Then, we have $P = !R$ and $Q = R \mid !R$. For part (1), suppose $\Gamma \vdash\ !R : T$. This could only be derived by using (ET Sub) and (ET Repl), with $\Gamma \vdash R : T_1$ and $T_1 \mid T_1 \leq T$. By (ET Par), we have $\Gamma \vdash R \mid !R : T_1 \mid_t (T_1 \mid_t T_1)$. By the definition of $\mid_t$, we know T_1 must be some simple type Y^Z. Again, by the definition of $\mid_t$, we get $Y^Z \mid_t Y^Z = Y^Z \mid_t (Y^Z \mid_t Y^Z)$. Finally, $\Gamma \vdash R \mid !R : T$ by (ET Sub). For part (2), suppose $\Gamma \vdash R \mid !R : T$. This could only be derived by using (ET Sub) and (ET Par), with $\Gamma \vdash R : T_1$, $\Gamma \vdash !R : T_2$, and $T_1 \mid_t T_2 \leq T$. By (ET Repl), we know that T_1 must be some simple type Y^Z. Based on this, we can apply (Par Zero T) and (Par Strict T) in Property 2 and get $T_2 \leq T_1 \mid_t T_2$. Finally, $\Gamma \vdash\ !R : T$ by (ET Sub).

(Struct Repl Zero) Then, we have $P = !\mathbf{0}$ and $Q = \mathbf{0}$. For part (1), suppose $\Gamma \vdash !\mathbf{0} : T$. This could only be derived by using (ET Sub) and (ET Repl), with $\Gamma \vdash \mathbf{0} : T_1$ and $T_1 \mid_t T_1 \leq T$. By the definition of $\mid_t$, we know T_1 must be some simple type Y^Z. Based on this, we can apply (Par Zero T) and (Par Strict T) in Property 2 and get $T_1 \leq T_1 \mid_t T_1$. Finally, $\Gamma \vdash \mathbf{0} : T$ by (ET Sub). For part (2), suppose $\Gamma \vdash \mathbf{0} : T$. This could only be derived by using (ET Sub) and (ET Inact), with $\Gamma \vdash \mathbf{0} : \underline{\vee}^0$ and $\underline{\vee}^0 \leq T$. By (ET Repl) and (ET Sub), we can derive $\Gamma \vdash\ \mathbf{0} : T$.

(Struct ϵ) Then, we have $P = \epsilon\,.\,Q$. For part (1), suppose $\Gamma \vdash \epsilon\,.\,Q : T$. This could only be derived by using (ET Sub) and (ET Act), with $\Gamma \vdash \epsilon : W$, $\Gamma \vdash Q : T'$, and $W(T') \leq T$. By (ET ϵ), we know $W = \$$. So $W(T') = T'$. By (ET Sub), we get $\Gamma \vdash Q : T$. For part (2), suppose $\Gamma \vdash Q : T$. By Lemma 6, we know $\Gamma \vdash \diamond$. Then, by (ET ϵ), we have $\Gamma \vdash \epsilon : \$$. By (ET Act), we get $\Gamma \vdash \epsilon\,.\,Q : \$(T)$, that is, $\Gamma \vdash \epsilon\,.\,Q : T$.

(Struct .) Then, we have $P = (M_1.M_2).R$ and $Q = M_1.(M_2.R)$. For part (1), suppose $\Gamma \vdash (M_1\,.\,M_2)\,.\,R : T$. This could only be derived by using (ET Sub) and (ET Act), with $\Gamma \vdash M_1\,.\,M_2 : W_1$, $\Gamma \vdash R : T'$, and $W_1(T') \leq T$. The former judgement could only be derive by using (ET Path), with $\Gamma \vdash M_1 : W_2$, $\Gamma \vdash M_2 : W_3$, and $W_1 = W_2(W_3)$. By (ET Act) twice, we get $\Gamma \vdash M_1\,.\,(M_2\,.\,R) : W_2(W_3(T'))$. Since $W_2(W_3(T')) = (W_2(W_3))(T') = W_1(T')$, we get $\Gamma \vdash M_1\,.\,(M_2\,.\,R) : T$ by (ET Sub). Part (2) follows by a similar analysis.

∎

A.3 Proof of Theorem 1

This subsection gives the proof of Theorem 1, based on the lemmas developed above. First goes another technical lemma.

Lemma 10 *Suppose* $\Gamma \vdash M : Y^Z \cdot_t \$$*, we have:*
(1) *If* $\Gamma \vdash M \,.\, P : T$*, then* $\Gamma \vdash P : T'$ *and* $T' \leq T$*;*
(2) *If* $\Gamma \vdash M \,.\, P \mid Q : T$*, then* $\Gamma \vdash P \mid Q : T'$ *and* $T' \leq T$*.*

Proof. We prove (1) first, then (2).

(1) The judgement $\Gamma \vdash M \,.\, P : T$ could only be derived by using (ET Sub) and (ET Act), with $\Gamma \vdash M : W$, $\Gamma \vdash P : T'$, and $W(T') \leq T$. Since $\Gamma \vdash M : Y^Z \cdot_t \$$, we have $Y^Z \cdot_t T' \leq T$. By Property 3, we can get $T' \leq Y^Z \cdot_t T'$. So $T' \leq T$ holds.

(2) The judgement $\Gamma \vdash M \,.\, P \mid Q : T$ could only be derived by using (ET Sub) and (ET Par), with $\Gamma \vdash M \,.\, P : T_1$, $\Gamma \vdash Q : T_2$, and $T_1 \mid_t T_2 \leq T$. By (1), we know that $\Gamma \vdash P : T_1'$ and $T_1' \leq T_1$. Let $T' = T_1 \mid T_2$, we have $T' \leq T$ and $\Gamma \vdash P \mid Q : T'$ by (ET Par). ∎

Theorem 1 (Subject reduction) If $\Gamma \vdash P : T$ and $P \longrightarrow Q$, then $\Gamma \vdash Q : T'$ and $T' \leq T$.

Proof. By induction on the derivation of $P \longrightarrow Q$.

(Red In) That is, if $\Gamma \vdash m[\texttt{in}\, n \,.\, P_1 \mid P_2] \mid n[\overline{\texttt{in}}\, m \,.\, Q_1 \mid Q_2] : T$, then $\Gamma \vdash n[m[P_1 \mid P_2] \mid Q_1 \mid Q_2] : T'$ and $T' \leq T$. We prove it below:

(1)	$\Gamma \vdash m[\texttt{in}\, n \,.\, P_1 \mid P_2] \mid n[\overline{\texttt{in}}\, m \,.\, Q_1 \mid Q_2] : T$	given
(2.1)	$\Gamma \vdash m[\texttt{in}\, n \,.\, P_1 \mid P_2] : T_1$	(1), by (ET Sub)
(2.2)	$\Gamma \vdash n[\overline{\texttt{in}}\, m \,.\, Q_1 \mid Q_2] : T_2$	+(ET Par) only
(2.3)	$T_1 \mid_t T_2 \leq T$	
(3.1)	$\Gamma \vdash m : T_m$	(2.1), by (ET Sub)
(3.2)	$\Gamma \vdash \texttt{in}\, n \,.\, P_1 \mid P_2 : T_m$	+(ET Amb) only
(3.3)	$\underline{\vee}^0 \leq T_1$	
(4.1)	$\Gamma \vdash n : T_n$	(2.2), by (ET Sub)
(4.2)	$\Gamma \vdash \overline{\texttt{in}}\, m \,.\, Q_1 \mid Q_2 : T_n$	+(ET Amb) only
(4.3)	$\underline{\vee}^0 \leq T_2$	
(5)	$\Gamma \vdash \texttt{in}\, n : \curvearrowright^1 \cdot_t \$$	(4.1)+(ET Cap $\curvearrowright$)
(6)	$\Gamma \vdash P_1 \mid P_2 : T_m$	(3.2)(5)+Lemma 10.(2) with (ET Sub)
(7)	$\Gamma \vdash m[P_1 \mid P_2] : \underline{\vee}^0$	(3.1)(6)+(ET Amb)
(8)	$\Gamma \vdash \overline{\texttt{in}}\, m : \underline{\vee}^1 \cdot_t \$$	(3.1)+(ET Cap $\underline{\vee}$)
(9)	$\Gamma \vdash Q_1 \mid Q_2 : T_n$	(4.2)(8)+Lemma 10.(2) with (ET Sub)
(10)	$\Gamma \vdash m[P_1 \mid P_2] \mid Q_1 \mid Q_2 : \underline{\vee}^0 \mid_t T_n$	(7)(9)+(ET Par)
(11)	$\Gamma \vdash m[P_1 \mid P_2] \mid Q_1 \mid Q_2 : T_n$	(10)+Property 2
(12)	$\Gamma \vdash n[m[P_1 \mid P_2] \mid Q_1 \mid Q_2] : \underline{\vee}^0$	(4.1)(11)+(ET Amb)

(13)	$\underline{\vee}^0 \leq T$	(2.3)(3.3)(4.3)+Property 2
(14)	(result holds)	(12)(13), let $T' = \underline{\vee}^0$

(Red Out) That is, if $\Gamma \vdash n[m[\mathtt{out}\ n \,.\, P_1 \mid P_2] \mid \overline{\mathtt{out}}\ m \,.\, Q_1 \mid Q_2] : T$, then $\Gamma \vdash m[P_1 \mid P_2] \mid n[Q_1 \mid Q_2] : T'$ and $T' \leq T$. We prove it below:

(1)	$\Gamma \vdash n[m[\mathtt{out}\ n \,.\, P_1 \mid P_2] \mid \overline{\mathtt{out}}\ m \,.\, Q_1 \mid Q_2] : T$	given
(2.1)	$\Gamma \vdash n : T_n$	(1), by (ET Sub)
(2.2)	$\Gamma \vdash m[\mathtt{out}\ n \,.\, P_1 \mid P_2] \mid \overline{\mathtt{out}}\ m \,.\, Q_1 \mid Q_2 : T_n$	+(ET Amb) only
(2.3)	$\underline{\vee}^0 \leq T$	
(3.1)	$\Gamma \vdash m[\mathtt{out}\ n \,.\, P_1 \mid P_2] : T_1$	(2.2), by (ET Sub)
(3.2)	$\Gamma \vdash \overline{\mathtt{out}}\ m \,.\, Q_1 \mid Q_2 : T_2$	+(ET Par) only
(3.3)	$T_1 \mid_t T_2 \leq T_n$	
(4.1)	$\Gamma \vdash m : T_m$	(3.1), by (ET Sub)
(4.2)	$\Gamma \vdash \mathtt{out}\ n \,.\, P_1 \mid P_2 : T_m$	+(ET Amb) only
(4.3)	$\underline{\vee}^0 \leq T_1$	
(5)	$T_2 \leq T_n$	(4.3)(3.3)+Property 2
(6)	$\Gamma \vdash \overline{\mathtt{out}}\ m \,.\, Q_1 \mid Q_2 : T_n$	(3.2)(5)+(ET Sub)
(7)	$\Gamma \vdash \overline{\mathtt{out}}\ m : \underline{\vee}^1 \cdot_t \$$	(4.1)+(ET Cap $\underline{\vee}$)
(8)	$\Gamma \vdash Q_1 \mid Q_2 : T_n$	(6)(7)+Lemma 10.(2) with (ET Sub)
(9)	$\Gamma \vdash n[Q_1 \mid Q_2] : \underline{\vee}^0$	(2.1)(8)+(ET Amb)
(10)	$\Gamma \vdash \mathtt{out}\ n : \curvearrowright^1 \cdot_t \$$	(2.1)+(ET Cap $\curvearrowright$)
(11)	$\Gamma \vdash P_1 \mid P_2 : T_m$	(4.2)(10)+Lemma 10.(2) with (ET Sub)
(12)	$\Gamma \vdash m[P_1 \mid P_2] : \underline{\vee}^0$	(4.1)(11)+(ET Amb)
(13)	$\Gamma \vdash m[P_1 \mid P_2] \mid n[Q_1 \mid Q_2] : \underline{\vee}^0$	(12)(9)+(ET Par)
(14)	(result holds)	(13)(2.3), let $T' = \underline{\vee}^0$

(Red Open) That is, if $\Gamma \vdash \mathtt{open}\ n \,.\, P \mid n[\overline{\mathtt{open}} \,.\, Q_1 \mid Q_2] : T$, then $\Gamma \vdash P \mid Q_1 \mid Q_2 : T'$ and $T' \leq T$. We prove it below:

(1)	$\Gamma \vdash \mathtt{open}\ n \,.\, P \mid n[\overline{\mathtt{open}} \,.\, Q_1 \mid Q_2] : T$	given
(2.1)	$\Gamma \vdash \mathtt{open}\ n \,.\, P : T_1$	(1), by (ET Sub)
(2.2)	$\Gamma \vdash n[\overline{\mathtt{open}} \,.\, Q_1 \mid Q_2] : T_2$	+(ET Par) only
(2.3)	$T_1 \mid_t T_2 \leq T$	
(3.1)	$\Gamma \vdash n : T_n$	(2.2), by (ET Sub)
(3.2)	$\Gamma \vdash \overline{\mathtt{open}} \,.\, Q_1 \mid Q_2 : T_n$	+(ET Amb) only
(3.3)	$\underline{\vee}^0 \leq T_2$	
(4)	$T_1 \leq T$	(3.3)(2.3)+Property 2

(5.1)	$\Gamma \vdash \overline{\texttt{open}}.Q_1 : T_{n1}$	(3.2), by (ET Sub)
(5.2)	$\Gamma \vdash Q_2 : T_{q2}$	+(ET Par) only
(5.3)	$T_{n1} \mid_t T_{q2} \leq T_n$	
(6.1)	$\Gamma \vdash \overline{\texttt{open}} : W$	(5.1), by (ET Sub)
(6.2)	$\Gamma \vdash Q_1 : T_{q1}$	+(ET Act) only
(6.3)	$W(T_{q1}) \leq T_{n1}$	
(7)	$W = \underline{\vee}^1[\$]$	(6.1), by (ET $\overline{\texttt{open}}$) only
(8)	$\underline{\vee}^1[T_{q1}] \mid_t T_{q2} \leq T_n$	(7)(6.3)(5.3)+Property 2
(9)	$\underline{\vee}^1 \mid_t T_{q2}[T_{q1} \mid_t T_{q2}] \leq T_n$	(8)+the definition of $\mid_t$
(10.1)	$T_n = Y^Z[T'_n]$	(9)+the definition of $\leq$
(10.2)	$T_{q1} \mid_t T_{q2} \leq T'_n$	
(11)	$\Gamma \vdash Q_1 \mid Q_2 : T_{q1} \mid_t T_{q2}$	(6.2)(5.2)+(ET Par)
(12)	$\Gamma \vdash Q_1 \mid Q_2 : T'_n$	(11)(10.2)+(ET Sub)
(13)	$\Gamma \vdash \texttt{open}\, n : \underline{\vee}^1 \cdot_t (T'_n \mid_t \$)$	(10.1)+(ET $\texttt{open}$)
(14.1)	$\Gamma \vdash P : T_p$	(2.1), by (ET Sub)
(14.2)	$\underline{\vee}^1 \cdot_t (T'_n \mid_t T_p) \leq T_1$	+(ET Act) only
(15)	$\Gamma \vdash P \mid Q_1 \mid Q_2 : T_p \mid T'_n$	(14.1)(12)+(ET Par)
(16)	$T_p \mid T'_n \leq T$	(14.2)(4)+Property 2, 3
(17)	(result holds)	(15)(16), let $T' = T_p \mid T'_n$

(Red Par) That is, if $\Gamma \vdash P \mid Q : T$ and $P \longrightarrow P'$, then $\Gamma \vdash P' \mid Q : T'$ and $T' \leq T$. We prove it below:

(1)	$\Gamma \vdash P \mid Q : T$	given
(2.1)	$\Gamma \vdash P : T_p$	(1), by (ET Sub)
(2.2)	$\Gamma \vdash Q : T_q$	+(ET Par) only
(2.3)	$T_p \mid_t T_q \leq T$	
(3)	$P \longrightarrow P'$	given
(4.1)	$\Gamma \vdash P' : T'_p$	(2.1)(3)+induction hypothesis
(4.2)	$T'_p \leq T_p$	
(5)	$\Gamma \vdash P' \mid Q : T_p \mid_t T_q$	(4.1)(4.2)+(ET Sub)(ET Par)
(6)	(result holds)	(5)(2.3), let $T' = T_p \mid_t T_q$

(Red Res) That is, if $\Gamma \vdash (\nu n : T_n)P : T$ and $P \longrightarrow P'$, then $\Gamma \vdash (\nu n : T_n)P' : T'$ and $T' \leq T$. We prove it below:

(1)	$\Gamma \vdash (\nu n : T_n)P : T$	given
(2.1)	$\Gamma,\ n : T_n \vdash P : T_1$	(1), by (ET Sub)
(2.2)	$T_1 \leq T$	+(ET Res) only
(3)	$P \longrightarrow P'$	given
(4.1)	$\Gamma,\ n : T_n \vdash P : T_2$	(2.1)(3)+induction hypothesis
(4.2)	$T_2 \leq T_1$	
(5)	$\Gamma \vdash (\nu n : T_n)P : T_1$	(4.1)(4.2)+(ET Res)(ET Sub)
(6)	(result holds)	(5)(2.2), let $T' = T_1$

(Red Amb) That is, if $\Gamma \vdash n[P] : T$ and $P \longrightarrow P'$, then $\Gamma \vdash n[P'] : T'$ and $T' \leq T$. We prove it below:

(1)	$\Gamma \vdash n[P] : T$	given
(2.1)	$\Gamma \vdash n : T_n$	(1), by (ET Sub)
(2.2)	$\Gamma \vdash P : T_n$	+(ET Amb) only
(2.3)	$\underline{\vee}^0 \leq T$	
(3)	$P \longrightarrow P'$	given
(4.1)	$\Gamma \vdash P' : T'_n$	(2.2)(3)+induction hypothesis
(4.2)	$T'_n \leq T_n$	
(5)	$\Gamma \vdash n[P'] : \underline{\vee}^0$	(4.1)(4.2)+(ET Sub)(ET Amb)
(6)	(result holds)	(5)(2.3), let $T' = \underline{\vee}^0$

(Red $\equiv$) That is, if $\Gamma \vdash P : T$, $P \equiv P'$, $P' \longrightarrow P''$, and $P'' \equiv P'''$, then $\Gamma \vdash P''' : T'$ and $T' \leq T$. We prove it below:

(1)	$\Gamma \vdash P : T$	given
(2)	$P \equiv P'$	given
(3)	$\Gamma \vdash P' : T$	(1)(2)+Lemma 9
(4)	$P' \longrightarrow P''$	given
(5.1)	$\Gamma \vdash P'' : T_1$	(3)(4)+induction hypothesis
(5.2)	$T_1 \leq T$	
(6)	$P'' \equiv P'''$	given
(7)	$\Gamma \vdash P''' : T_1$	(5)(6)+Lemma 9
(8)	(result holds)	(7)(5.2), let $T' = T_1$

∎