Security Types for Mobile Safe Ambients *

M. Dezani-Ciancaglini, I. Salvo

Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 Torino, Italy
{dezani,salvo}@di.unito.it

Abstract. The *Ambient Calculus* and the *Safe Ambient Calculus* have been recently successfully proposed as models for the Web. They are based on the notions of ambient *movement* and ambient *opening*. Different type disciplines have been devised for them in order to avoid unwanted behaviours of processes.
In the present paper we propose a type discipline for safe mobile ambients which is essentially motivated by ensuring *security* properties. We associate security levels to ambients and we require that an ambient at security level s can only be traversed or opened by ambients at security level at least s. Since the movement and opening rights can be unrelated, we consider two partial orders between security levels.
We also discuss some meaningful examples of use of our type discipline.

1 Introduction

The *Ambient Calculus* [4] has been recently successfully proposed as a model for the Web. An ambient is a named location: it may contain processes and sub-ambients. A process may:

- communicate in an asynchronous way with a process in the same ambient;
- cause the enclosing ambient to move inside or outside other ambients;
- destroy the boundary of a sub-ambient, causing the contents of the sub-ambient to spill into the parent ambient.

In order to have a richer algebraic theory, in the *Safe Ambient Calculus* [9] the activity of processes is better controlled since:

- an ambient may traverse an ambient n only if at least one process inside n agrees;
- a process may destroy the boundary of an ambient n only if at least one process inside n agrees.

A standard way of forbidding unwanted behaviours is to impose a *type discipline*.

Different type disciplines have been proposed for the Ambient Calculus, taking advantage of several papers on typing for mobile processes, with or without localities, see for example [11], [6] and [10]. In [5] the types assure the correctness of communications. The type system of [2] guarantees also that only ambients

* Partially supported by MURST Cofin '99 TOSCA Project.

which are declared as mobile will move and only ambients which are declared as openable will be opened. Adding subtyping allows us to obtain a more flexible type discipline [12]. Lastly, by means of *group* names [3], the type of an ambient n controls the set of ambients n may cross and the set of ambients n may open. Moreover the possibility of creating fresh group names gives a flexible way of statically preventing unwanted propagation of names.

A powerful type discipline for the Safe Ambient Calculus has been devised in [9]. The main features are the control of ambient mobility and the removing of all *grave interferences*, i.e. of all non-deterministic choices between logically incompatible interactions. This is achieved by means of types which can be derived only for *single-threaded* ambients, i.e. ambients which at every step offer at most one interaction with external or internal ambients.

The secure safe ambient calculus of [1] is a typed variant of Safe Ambients in which ambient types are protection domains expressing behavioural invariants.

Security plays a crucial role in the theory and practice of distributed systems. In the present paper we propose a type discipline for safe mobile ambients which is essentially motivated by ensuring *security* properties. The type of an ambient name specifies a security level s. We require that an ambient at security level s can only be traversed or opened by ambients at security level at least s. For example the movement of an ambient n inside an ambient m:

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

can correctly be typed only if the security level of n is greater than or equal to that of m. Moreover, we consider also *passive* security levels, which forbid an ambient to influence the behaviour of a surrounding ambient belonging to a higher security level.

Since the movement and opening rights can be unrelated, we consider two partial orders between security levels.

As in the above-mentioned type discipline with group names, each ambient name belongs to a security level. However, we do not consider the possibility of creating fresh, private security levels, since we consider two global partial orders defined on security levels and it is not clear how to include the new level. Moreover, thanks to the order relations, in the type of an ambient name we do not need to list explicitly all security levels of ambients it may traverse or open.

In the simple case in which the movement rights are all equal and there are only two opening rights we classify the ambients as trustworthy and untrustworthy. Then the types can assure the secrecy property considered in [8] that an untrustworthy ambient can never open a trustworthy one.

The paper is organized as follows. Section 2 recalls the definitions of Mobile Safe Ambients. For the sake of readability, we present our type system in two steps. Section 3 discusses a simpler version, which introduces the notion of security level in ambient types. The full system is motivated by the requirement of obtaining more refined typings, in particular to type as immobile an ambient which opens a mobile ambient that, when opened, does not unleash mobility capabilities. To do this we need to distinguish the behaviours of processes before and after an ambient is opened (Section 4). Section 5 gives some examples of use of our type discipline. A first protocol models a mailserver with different mailboxes and users: each user is allowed to enter only his own mailbox and this is achieved via type constraints imposed on the security level order. A second example shows that we can encode the security policy for reading and writing discussed in [7] for the π-calculus. Lastly, we present the renaming, firewall and channel protocols which are already typed in [2] and in [9] for comparisons and for showing how some behavioural conditions can be expressed in our system as type constraints. Some final remarks are done in Section 6.

2 Untyped Mobile Safe Ambients

The calculus of Mobile Safe Ambients [9] is a refinement of the calculus of Mobile Ambients [4] which allows a better control of the actions and therefore a richer algebraic theory of processes. We will use here the calculus of Mobile Safe Ambients, the only difference being that we describe infinite behaviours by $!P$ instead of $\mathbf{rec}X.P$, since this slightly simplifies the typing rules. The syntax is given in Figure 1 starting from an infinite countable set of names.

Figure 2 contains the reduction relation which uses the structural congruence $\equiv$. As customary, the structural congruence $\equiv$ is defined as the minimal reflexive, transitive and symmetric relation which is a congruence and moreover:

- satisfies $!P \equiv !P \mid P$;
- makes the operator $\mid$ commutative, associative, with $\mathbf{0}$ as zero element;
- allows to stretch the scopes of restrictions, to permute restrictions and to cancel a restriction followed only by $\mathbf{0}$;
- satisfies $\epsilon.P \equiv P$ and $(M.M').P \equiv M.M'.P$.

$M, N ::=$	**expression**	$P, Q ::=$	**process**
n	name	$(\nu\, n : W)P$	restriction
$\mathbf{in}\ M$	can inter into M	$\mathbf{0}$	inactivity
$\overline{\mathbf{in}}\ M$	M allow enter	$P \mid Q$	parallel
$\mathbf{out}\ M$	can exit out of M	$!P$	replication
$\overline{\mathbf{out}}\ M$	M allow exit	$M.P$	action
$\mathbf{open}\ M$	can open M	$M[P]$	ambient
$\overline{\mathbf{open}}\ M$	M allow open	$\langle M_1 \dots M_k \rangle$	output action
ϵ	empty path	$(x_1 : W_1, \dots, x_k : W_k).P$	input action
$M.M'$	path		

Fig. 1. Expressions and Processes

3 Security Types

The basic idea of our type system is to control at type level access and opening rights of ambients. Each ambient name has a type that represents its *security*

$$
\begin{array}{lll}
\text{(R-in)} & n[\mathbf{in}\ m.P_1 \mid P_2] \mid m[\overline{\mathbf{in}}\ m.Q_1 \mid Q_2] & \longrightarrow m[n[P_1 \mid P_2] \mid Q_1 \mid Q_2] \\
\text{(R-out)} & m[n[\mathbf{out}\ m.P_1 \mid P_2] \mid \overline{\mathbf{out}}\ m.Q_1 \mid Q_2] & \longrightarrow n[P_1 \mid P_2] \mid m[Q_1 \mid Q_2] \\
\text{(R-open)} & \mathbf{open}\ n.P \mid n[\overline{\mathbf{open}}\ n.Q_1 \mid Q_2] & \longrightarrow P \mid Q_1 \mid Q_2 \\
\text{(R-I/O)} & \langle M_1, \ldots, M_k \rangle \mid (n_1 : W_1, \ldots, n_k : W_k).P & \longrightarrow P\{n_1 := M_1, \ldots, n_k := M_k\} \\
\text{(R-par)} & P \longrightarrow Q & \Rightarrow\ P \mid R \longrightarrow Q \mid R \\
\text{(R-res)} & P \longrightarrow Q & \Rightarrow\ (\nu n : W)P \longrightarrow (\nu n : W)Q \\
\text{(R-amb)} & P \longrightarrow Q & \Rightarrow\ n[P] \longrightarrow n[Q] \\
\text{(R-}\equiv\text{)} & P' \equiv P \quad P \longrightarrow Q \quad Q \equiv Q' & \Rightarrow\ P' \longrightarrow Q'
\end{array}
$$

Fig. 2. Reduction

level: a process inside an ambient can perform actions according to the ambient security level. A security level defines mobility and opening rights of ambients which belong to. As in previous type systems for Mobile Ambients [5, 2], the type of an ambient keeps information about the type of messages exchanged inside it, if the ambient can move and if it can be opened. Moreover, in the type of an ambient name, we consider also type annotations that say if the ambient can be traversed and if it can open other ambients. In Section 5, we successfully exploit security levels to model a mailserver where unauthorized accesses are forbidden as non-well typed processes (Subsection 5.1) and to model an information flow policy in an ambient encoding of π-calculus channels.

Types. Types are defined starting from an universe $\mathcal{U}$ of *active security levels* and an universe $\overline{\mathcal{U}}$ of *passive* security levels: each element s in $\mathcal{U}$ has a corresponding element $\bar{\mathsf{s}}$ in $\overline{\mathcal{U}}$. The universe $\mathcal{U}$ comes equipped with two partial order relations, $\leq^{\frown}$ for mobility rights and $\leq^{\circ}$ for opening rights. These orders are mirrored in $\overline{\mathcal{U}}$.

We denote by $\mathsf{s}, \mathsf{s}', \mathsf{s}_1, \ldots$ elements of $\mathcal{U}$, by $\bar{\mathsf{s}}, \bar{\mathsf{s}}', \bar{\mathsf{s}}_1, \ldots$ elements of $\overline{\mathcal{U}}$, and by $\mathsf{S}, \mathsf{S}', \mathsf{S}_1, \ldots$ subsets of $\mathcal{U} \cup \overline{\mathcal{U}}$.

As usual [5] the exchange types are Shh when no exchange is allowed and tuples whose elements are either ambient types or capability types.

An ambient type has the shape $\mathsf{s}_V^Y[T_U^Z]$ where:

- s is the security level to which the ambient belongs (similar to the group of [3]);
- T is the type of exchanges the ambient allows within (as in [5]);
- Y says if the ambient is locked ($\bullet$) or unlocked ($\circ$) and only unlocked ambients can be opened (as in [2]);
- V says if the ambient can ($\bar{\circ}$) or cannot ($\bar{\bullet}$) open another ambient which belongs to a security level lower or equal in the order $\leq^{\circ}$;
- Z says if the ambient can move ($\frown$) or if it is immobile ($\underline{\vee}$) (as in [2]);
- U says if the ambient can ($\leftrightarrow$) or cannot ($\oplus$) be traversed by another ambient which belongs to a security level greater or equal in the order $\leq^{\frown}$.

A capability type has the shape $Cap[F]$ where F is an effect.

An effect has the shape $\mathsf{S}_1, \mathsf{S}_2, T$, where T is the type of exchanges which can be unleashed by an **open** action (as in [3]). The set S_1 contains the security levels of the ambients which can be traversed. The set S_2 contains the security levels of the ambients which can be opened. In S_1 and S_2 the levels are active or passive according to the ambient is in an action or in a co-action.

The sets of type annotations and of types are respectively given in Figures 3 and 4.

$Z :=$	**mobility annotation**	$V :=$	**opening annotation**
	$\curvearrowright$ mobile		$\overline{\circ}$ can open
	$\underline{\vee}$ immobile		$\overline{\bullet}$ cannot open
$U :=$	**traversing annotation**	$Y :=$	**locking annotation**
	$\leftrightarrow$ can be traversed		$\circ$ unlocked
	$\oplus$ cannot be traversed		$\bullet$ locked

Fig. 3. Type annotations

F ::=		**effect**
	$\mathsf{S}_1, \mathsf{S}_2, T$	moves and is traversed according to S_1, open and can be opened according to S_2, exchange T
W ::=		**message type**
	$\mathsf{s}_V^Y[T_U^Z]$	ambient name in security level s (contains processes whose effects agree with $\{Y, V, Z, U\}$)
	$Cap[F]$	capability (unleashes F effects)
T ::=		**exchange type**
	Shh	no exchange
	$W_1 \times \ldots \times W_k$	tuple exchange

Fig. 4. Types

Typing Rules. As usual an environment E associates names with ambient and capability types:

$$E ::= \varnothing \mid E, n : W.$$

The domain of E (notation $dom(E)$) is defined by:

$$dom(E) = \begin{cases} \varnothing & \text{if } E = \varnothing \\ dom(E') \cup \{n\} & \text{if } E = E', n : W \end{cases}$$

The typing judgments are relative to a given universe $\mathcal{U}$ of security levels and to a given environment E. There are six kinds of judgments:

$\mathcal{U}, E \vdash \diamond$	good environment
$\mathcal{U}, E \vdash T$	good exchange type T
$\mathcal{U}, E \vdash F$	good effect F
$\mathcal{U}, E \vdash W$	good message type W
$\mathcal{U}, E \vdash M : W$	good expression M of message type W
$\mathcal{U}, E \vdash P : F$	good process P with effect F.

Figure 5 contains the rules for deriving judgments of the four first kinds. These rules follow in a standard way the syntax of types given in Figure 4.

$$\frac{}{\mathcal{U} \vdash \diamond}\text{(Empty Env)} \qquad \frac{\mathcal{U}, E \vdash W \quad n \notin dom(E)}{\mathcal{U}, E, n : W \vdash \diamond}\text{(Env Formation)}$$

$$\frac{\mathcal{U}, E \vdash \diamond}{\mathcal{U}, E \vdash Shh}(Shh) \qquad \frac{\mathcal{U}, E \vdash W_1 \quad \ldots \quad \mathcal{U}, E \vdash W_k}{\mathcal{U}, E \vdash W_1 \times \ldots \times W_k}\text{(Prod)}$$

$$\frac{\mathcal{U}, E \vdash T \quad \mathsf{s} \in \mathcal{U}}{\mathcal{U}, E \vdash \mathsf{s}_V^Y[T_U^Z]}\text{(Amb)} \qquad \frac{\mathcal{U}, E \vdash F}{\mathcal{U}, E \vdash Cap[F]}\text{(Cap)}$$

$$\frac{\mathcal{U}, E \vdash T \quad \bigcup_{1 \leq i \leq 2} \mathsf{S}_i \subseteq \mathcal{U} \cup \overline{\mathcal{U}}}{\mathcal{U}, E \vdash \mathsf{S}_1, \mathsf{S}_2, T}\text{(Effect)}$$

Fig. 5. Good Environments and Types

The capabilities unleashed by opening an ambient n are the maximal capabilities consistent with respect to the type annotations and the security level in the type of n, therefore to give the typing rules for the **open** expression we need to define when *two sets* S, S' *of security levels* agree ($\bowtie$) *with the type annotation* $\{V, Z, U\}$. This means that:

- if S contains an active security level then Z says that the ambient is mobile;
- if S contains a passive security level then U says that the ambient can be traversed;
- if S' contains an active security level then V says that the ambient can open.

This allows us to build the pair of maximal sets $\Im(\{V, Z, U\}, \mathsf{s})$ which agree with a given type annotation $\{V, Z, U\}$ and contain only a given security level (active) s and (passive) $\overline{\mathsf{s}}$. See Figure 6.

$$\mathsf{S}, \mathsf{S}' \bowtie \{V, Z, U\} \iff \begin{array}{l} \exists \mathsf{s} \in \mathsf{S} \;\Rightarrow\; Z = \curvearrowright \;\; \& \\ \exists \overline{\mathsf{s}} \in \mathsf{S} \;\Rightarrow\; U = \leftrightarrow \;\; \& \\ \exists \mathsf{s} \in \mathsf{S}' \;\Rightarrow\; V = \overline{\circ} \end{array}$$

$$\Im(\{V, Z, U\}, \mathsf{s}) = max\ \mathsf{S}, \mathsf{S}'.\mathsf{S} \cup \mathsf{S}' \subseteq \{\mathsf{s}, \overline{\mathsf{s}}\} \;\;\&\;\; \mathsf{S}, \mathsf{S}' \bowtie \{V, Z, U\}$$

Fig. 6. Agreement between sets of security levels and type annotations

To determine the effect of a path of expressions, we combine two effects for obtaining a new effect. As usual, we can combine two effects only if they share the same exchange type. The *combination of effects* is just componentwise set union, more precisely (overloading the symbol $\cup$):

$$(\mathsf{S}_1, \mathsf{S}_2, T) \cup (\mathsf{S}_3, \mathsf{S}_4, T) = \mathsf{S}_1 \cup \mathsf{S}_3, \mathsf{S}_2 \cup \mathsf{S}_4, T.$$

The typing of a mobility action, **in** M or **out** M, checks if the ambient M can be traversed and builds a simple capability by putting the active security level of M in the first set. The typing of a mobility co-action, $\overline{\textbf{in}}$ M or $\overline{\textbf{out}}$ M, is similar, except that the passive security level is put in the first set.

The typing of an open action, **open** M, checks if the ambient M can be opened, using the information given by the type of M, that has the shape $\mathsf{s}_V^Y[T_U^Z]$. If Y is $\circ$, the ambient can be opened and in the type of **open** M we must take into account the effects unleashed by the ambient M after being opened. This effect must contain a pair of sets S, S' which agree with $\{V, Z, U\}$. Since we have to take into account the maximal possible effect, we build S, S' as $\Im(\{V, Z, U\}, \mathsf{s})$. The final effect will be $\mathsf{S}, \{\mathsf{s}\} \cup \mathsf{S}', T$, where we add s to take into account the open action.

An open co-action, $\overline{\textbf{open}}$ M, can be typed only if the annotation Y in the type of M is $\circ$ and the effect is simply $\{\ \}, \{\bar{\mathsf{s}}\}, T$. Figure 7 gives the typing rules for expressions.

Typing an ambient requires to check if the ambient type agrees with the effect of the process running inside it. An *effect* $\mathsf{S}, \mathsf{S}', T$ agrees *with an ambient type* $\mathsf{s}_V^Y[T_U'^Z]$ iff:
- $T = T'$;
- the sets S, S' agree with the type annotations $\{V, Z, U\}$;
- if S' contains a passive security level then Y says that the ambient can be opened;
- all active security levels in S are $\leq^{\frown}$ s and all passive security levels in S are $\leq^{\frown}$ $\bar{\mathsf{s}}$;
- all active security levels in S' are $\leq^{\circ}$ s and all passive security levels in S' are $\leq^{\circ}$ $\bar{\mathsf{s}}$.

Conditions on passive security levels ensure that opening an ambient does not influence, unleashing co-capabilities, the behaviour of a surrounding ambient which belongs to a higher security level.

In Figure 8 we formally define when an effect $\mathsf{S}, \mathsf{S}', T$ agrees with $\mathsf{s}_V^Y[T_U'^Z]$ using the definitions of Figure 6.

Figure 9 gives the typing rules for processes. Rule (Proc Amb) checks that the effect of P agrees with the ambient type of n before building the process $n[P]$. Now $n[P]$ has no action, and therefore its effect is $\{\ \}, \{\ \}, T$ for an arbitrary T. The remaining typing rules are almost standard.

Properties of Well-Typed Processes. The effects we derive for processes are rather informative of their behaviours. More precisely if $\mathcal{U}, E \vdash P : \mathsf{S}, \mathsf{S}', T$ we can show that:
- if P may perform a move action then $\mathsf{S} \cap \mathcal{U} \neq \varnothing$,
- if P may perform a move co-action then $\mathsf{S} \cap \overline{\mathcal{U}} \neq \varnothing$,
- if P may perform an open action then $\mathsf{S}' \cap \mathcal{U} \neq \varnothing$,
- if P may perform an open co-action then $\mathsf{S}' \cap \overline{\mathcal{U}} \neq \varnothing$.

It turns out that a process P has no thread [9], i.e. P cannot do actions, whenever we can derive the judgement $\mathcal{U}, E \vdash P : \mathsf{S}, \mathsf{S}', T$ with $(\mathsf{S} \cup \mathsf{S}') \cap \mathcal{U} = \varnothing$.

$$\frac{\mathcal{U}, E, n : W, E' \vdash \diamond}{\mathcal{U}, E, n : W, E' \vdash n : W} \text{(Exp } n\text{)} \qquad \frac{\mathcal{U}, E \vdash \diamond}{\mathcal{U}, E \vdash \epsilon : \{\}, \{\}, T} \text{(Exp } \epsilon\text{)}$$

$$\frac{\mathcal{U}, E \vdash M : Cap[F] \quad \mathcal{U}, E \vdash M' : Cap[F']}{\mathcal{U}, E \vdash M.M' : Cap[F \cup F']} \text{(Exp .)}$$

$$\frac{\mathcal{U}, E \vdash M : \mathsf{s}_V^Y[T_{\leftrightarrow}^Z] \quad \mathcal{U}, E \vdash T'}{\mathcal{U}, E \vdash \mathbf{in}\ M : Cap[\{\mathsf{s}\}, \{\ \}, T']} \text{(Exp } \mathbf{in}\text{)}$$

$$\frac{\mathcal{U}, E \vdash M : \mathsf{s}_V^Y[T_{\leftrightarrow}^Z] \quad \mathcal{U}, E \vdash T'}{\mathcal{U}, E \vdash \overline{\mathbf{in}}\ M : Cap[\{\overline{\mathsf{s}}\}, \{\ \}, T']} \text{(Exp } \overline{\mathbf{in}}\text{)}$$

$$\frac{\mathcal{U}, E \vdash M : \mathsf{s}_V^Y[T_{\leftrightarrow}^Z] \quad \mathcal{U}, E \vdash T'}{\mathcal{U}, E \vdash \mathbf{out}\ M : Cap[\{\mathsf{s}\}, \{\ \}, T']} \text{(Exp } \mathbf{out}\text{)}$$

$$\frac{\mathcal{U}, E \vdash M : \mathsf{s}_V^Y[T_{\leftrightarrow}^Z] \quad \mathcal{U}, E \vdash T'}{\mathcal{U}, E \vdash \overline{\mathbf{out}}\ M : Cap[\{\overline{\mathsf{s}}\}, \{\ \}, T']} \text{(Exp } \overline{\mathbf{out}}\text{)}$$

$$\frac{\mathcal{U}, E \vdash M : \mathsf{s}_V^{\circ}[T_U^Z] \quad \Im(\{V, Z, U\}, \mathsf{s}) = \mathsf{S}, \mathsf{S}'}{\mathcal{U}, E \vdash \mathbf{open}\ M : Cap[\mathsf{S}, \{\mathsf{s}\} \cup \mathsf{S}', T]} \text{(Exp } \mathbf{open}\text{)}$$

$$\frac{\mathcal{U}, E \vdash M : \mathsf{s}_V^{\circ}[T_U^Z]}{\mathcal{U}, E \vdash \overline{\mathbf{open}}\ M : Cap[\{\ \}, \{\overline{\mathsf{s}}\}, T]} \text{(Exp } \overline{\mathbf{open}}\text{)}$$

Fig. 7. Good Expressions

$$\mathsf{S}, \mathsf{S}', T \bowtie \mathsf{s}_V^Y[T_U'^Z] \iff \begin{array}{l} T = T'\ \& \\ \mathsf{S}, \mathsf{S}' \bowtie \{V, Z, U\}\ \& \\ \exists \overline{s} \in \mathsf{S}' \ \Rightarrow\ Y = \circ\ \& \\ \forall \mathsf{s}', \overline{\mathsf{s}}' \in \mathsf{S}.\mathsf{s}' \leq^{\frown} \mathsf{s}\ \&\ \overline{\mathsf{s}}' \leq^{\frown} \overline{\mathsf{s}}\ \& \\ \forall \mathsf{s}', \overline{\mathsf{s}}' \in \mathsf{S}'.\mathsf{s}' \leq^{\circ} \mathsf{s}\ \&\ \overline{\mathsf{s}}' \leq^{\circ} \overline{\mathsf{s}} \end{array}$$

Fig. 8. Agreement between effects and ambient types

$$\frac{\mathcal{U}, E \vdash M : Cap[F] \quad \mathcal{U}, E \vdash P : F'}{\mathcal{U}, E \vdash M.P : F \cup F'} \text{(Proc Action)}$$

$$\frac{\mathcal{U}, E \vdash M : \mathsf{s}_V^Y[T_U^Z] \quad \mathcal{U}, E \vdash P : F \quad F \bowtie \mathsf{s}_V^Y[T_U^Z] \quad \mathcal{U}, E \vdash T}{\mathcal{U}, E \vdash M[P] : \{\ \}, \{\ \}, T} \text{(Proc Amb)}$$

$$\frac{\mathcal{U}, E, n : \mathsf{s}_V^Y[T_U^Z] \vdash P : F}{\mathcal{U}, E \vdash (\nu\, n : \mathsf{s}_V^Y[T_U^Z])P : F} \text{(Proc Res)} \qquad \frac{\mathcal{U}, E \vdash T}{\mathcal{U}, E \vdash \mathbf{0} : \{\ \}, \{\ \}, T} \text{(Proc } \mathbf{0}\text{)}$$

$$\frac{\mathcal{U}, E \vdash P : F \quad \mathcal{U}, E \vdash Q : F'}{\mathcal{U}, E \vdash P \mid Q : F \cup F'} \text{(Proc Par)} \qquad \frac{\mathcal{U}, E \vdash P : F}{\mathcal{U}, E \vdash !P : F} \text{(Proc Repl)}$$

$$\frac{\mathcal{U}, E, n_1 : W_1, \ldots, n_k : W_k \vdash P : \mathsf{S}, \mathsf{S}', W_1 \times \ldots \times W_k}{\mathcal{U}, E \vdash (n_1 : W_1, \ldots, n_k : W_k).P : \mathsf{S}, \mathsf{S}', W_1 \times \ldots \times W_k} \text{(Proc Input)}$$

$$\frac{\mathcal{U}, E \vdash M_1 : W_1 \quad \ldots \quad \mathcal{U}, E \vdash M_k : W_k}{\mathcal{U}, E \vdash \langle M_1, \ldots, M_k \rangle : \{\ \}, \{\ \}, W_1 \times \ldots \times W_k} \text{(Proc Output)}$$

Fig. 9. Good Processes

This implies that we can type only single threaded processes if we restrict rule (Proc Par) as follows:

$$\frac{\mathcal{U}, E \vdash P : F \quad \mathcal{U}, E \vdash Q : F' \quad st(F) \text{ or } st(F')}{\mathcal{U}, E \vdash P \mid Q : F \cup F'}$$

where $st(\mathsf{S}, \mathsf{S}', T)$ is short for $(\mathsf{S} \cup \mathsf{S}') \cap \mathcal{U} = \varnothing$. A more refined typing for single threaded processes is given in [9].

As usual, the soundness of the typing rules is assured by a Subject Reduction Theorem. Since the effects precisely describe the actions which can be performed by a process, and the reduction consumes actions, in general we can derive a "better" effect for the reduced process. To formally state this, we introduce an order $\sqsubseteq$ on effects: $F' \sqsubseteq F$ means that sets in F "cover" with respect to the appropriate order the corresponding sets in F'. We say that a set S' *covers* a set S with respect to the order $\leq^*$ ($\mathsf{S} \sqsubseteq^* \mathsf{S}'$) iff for all $\mathsf{s} \in \mathsf{S}$ we can find $\mathsf{s}' \in \mathsf{S}'$ such that $\mathsf{s} \leq^* \mathsf{s}'$ and similarly for passive security levels. Figure 10 defines this order.

$$\begin{array}{lcl} \mathsf{S} \sqsubseteq^* \mathsf{S}' & \iff & \forall \mathsf{s} \in \mathsf{S}. \exists \mathsf{s}' \in \mathsf{S}'. \mathsf{s} \leq^* \mathsf{s}' \ \&\ \forall \bar{\mathsf{s}} \in \mathsf{S}. \exists \bar{\mathsf{s}}' \in \mathsf{S}'. \bar{\mathsf{s}} \leq^* \bar{\mathsf{s}}' \quad (* \in \{\frown, \circ\}) \\ \mathsf{S}_1, \mathsf{S}_2, T \sqsubseteq \mathsf{S}_1', \mathsf{S}_2', T & \iff & \mathsf{S}_1 \sqsubseteq^{\frown} \mathsf{S}_1' \ \&\ \mathsf{S}_2 \sqsubseteq^{\circ} \mathsf{S}_2' \end{array}$$

Fig. 10. Partial order of effects

Theorem 1. *If* $\mathcal{U}, E \vdash P : F$ *and* $P \longrightarrow Q$ *then* $\mathcal{U}, E \vdash Q : F'$ *for some* F' *such that* $F' \sqsubseteq F$.

We will prove the soundness of the full system (Theorem 2) which implies that of the present one.

The subject reduction property means that:

- every communication is well-typed;
- no locked ambient will ever be opened, and an unlocked ambient will only be opened inside an ambient which can open and which has a higher or equal security level in the order $\leq^{\circ}$;
- no immobile ambient will ever move, and a mobile ambient will only traverse ambients which can be traversed and which have lower or equal security levels in the order $\leq^{\frown}$.

4 Security Types with Opening Control

As pointed out in [9], better control over the ambient behaviours helps in deriving a richer algebraic theory and hence easier proofs of correctness.

One important lack of the type system introduced in previous section, with respect of the type system of [9], is that it is not possible to type as immobile an ambient that opens mobile ambients. This is not satisfactory. As an example think to a message sent to a server: the message is a mobile entity, whereas the server is not. Reasonably, in an Ambient Calculus encoding, the ambient representing the server has to open the ambient representing the message. A message is represented by a mobile ambient, but opening it does not unleash movement capabilities. We will further discuss this problem in the example concerning a firewall protocol, in Subsection 5.3.

In this section we extend our type system in order to derive such kind of judgements. In the typical cases in which only one $\overline{\mathbf{open}}$ appears in processes inside an ambient, exploiting the presence of co-actions, we can distinguish between effects before and after the unique $\overline{\mathbf{open}}$ co-action and hence determine the capabilities unleashed opening such an ambient. As expected, such control over ambient behaviour rather complicates the type system.

Expanded Types. We extend the set of types as follows. The locking annotations can also have the shape:

$$Y = \circ\{V, Z, U\}$$

which means that the ambient can be opened and that processes spilled out after the opening behave according to the annotation $\{V, Z, U\}$.

Also effects can keep information about opening and mobility rights required by processes before and after the opening of their surrounding ambient. Therefore effects can be *simple effects* that, as before, have the shape $\mathsf{S}_1, \mathsf{S}_2, T$ or *expanded effects*, that have the shape $\mathsf{S}_1, \mathsf{S}_2, \{\mathsf{S}_3, \mathsf{S}_4\}, T$ which says that there is only one $\overline{\mathbf{open}}$ and that before the execution of it we have the effect $\mathsf{S}_1, \mathsf{S}_2, T$, while after the execution of it we have the effect $\mathsf{S}_3, \mathsf{S}_4, T$. In Fig. 11 we extend the syntax of effects.

$H ::=$		**opening effect**
	S	open and can be opened according to S
	$\mathsf{S}_1, \{\mathsf{S}_2, \mathsf{S}_3\}$	open and can be opened according to S_1 after the open, moves and is traversed according to S_2, open and can be opened according to S_3
$F ::=$		**effect**
	S, H, T	moves and is traversed according to S, open and can be opened according to H, exchange T

Fig. 11. Expanded Effects

Expanded Typing Rules. The rules for type formation must be extended by adding a rule for the formation of expanded effects:

$$\frac{\mathcal{U}, E \vdash T \quad \bigcup_{1 \leq i \leq 4} \mathsf{S}_i \subseteq \mathcal{U} \cup \overline{\mathcal{U}} \quad \mathsf{S}_2 \cap \overline{\mathcal{U}} \neq \varnothing}{\mathcal{U}, E \vdash \mathsf{S}_1, \mathsf{S}_2, \{\mathsf{S}_3, \mathsf{S}_4\}, T} \text{(Effect expanded)}$$

where we require that at least one ambient can be opened with the condition $\mathsf{S}_2 \cap \overline{\mathcal{U}} \neq \varnothing$.

The combination of two effects is still componentwise set union when the first one is simple, but now the resulting effect is either simple or expanded according to how the second effect is. If the effects are both expanded their combination is a simple effect obtained by loosing the information on what is the difference between the effects before and after the two $\overline{\textbf{open}}$ actions: in such case we can not foresee, typing an **open** action, which $\overline{\textbf{open}}$ will be involved in the reduction, and hence the extra information given by expanded effects is not useful to obtain more refined typings. If the first effect is expanded and the second is simple, the combination is an expanded effect obtained by taking into account the second effect only after the unique $\overline{\textbf{open}}$ action. The formal definition of combination of effects is given in Figure 12.

$$\mathsf{S}, H, T \uplus \mathsf{S}', H', T' = \begin{cases} \mathsf{S} \cup \mathsf{S}', H \cup H', T & \text{if } H \subseteq \mathcal{U} \cup \overline{\mathcal{U}} \\ \mathsf{S}, \mathsf{S}_1, \{\mathsf{S}_2 \cup \mathsf{S}', \mathsf{S}_3 \cup H'\}, T & \text{if } H = \mathsf{S}_1, \{\mathsf{S}_2, \mathsf{S}_3\} \ \& \\ & H' \subseteq \mathcal{U} \cup \overline{\mathcal{U}} \\ \mathsf{S} \cup \mathsf{S}' \cup \mathsf{S}_2 \cup \mathsf{S}_2', \mathsf{S}_1 \cup \mathsf{S}_1' \cup \mathsf{S}_3 \cup \mathsf{S}_3', T & \text{if } H = \mathsf{S}_1, \{\mathsf{S}_2, \mathsf{S}_3\} \ \& \\ & H' = \mathsf{S}_1', \{\mathsf{S}_2', \mathsf{S}_3'\} \end{cases}$$

Fig. 12. Combination of simple and expanded effects for expressions

In typing rules for expression we have to modify the rule for expression path (Exp .), replacing the operator $\uplus$ to set union for combination of effects. The other rules remain the same, also if Y can be $\circ\{V, Z, U\}$, and we add rules for all actions to take into account expanded effects. The new rules for typing the mobility actions allow the movement after the ambient has been opened.

Similarly to the case of simple effects, the typing of an open action, **open** M, checks if the ambient M can be opened, using the information given by the type of M, that has the shape $\mathsf{s}_V^Y[T_U^Z]$. If Y is $\circ\{V', Z', U'\}$, the ambient can be opened and in the type of **open** M we must take into account the effects unleashed by the ambient M after being opened. This effect must contain a pair of sets S, S' which agree with $\{V', Z', U'\}$, so as for simple effects we take $\Im(\{V', Z', U'\}, \mathsf{s})$. The final effect will be $\mathsf{S}, \{\mathsf{s}\} \cup \mathsf{S}', T$, where we add s to take into account the open action.
An open co-action, $\overline{\mathbf{open}}$ M, can be typed if the annotation Y in the type of M is $\circ\{V', Z', U'\}$. In this case the effect is expanded to $\{\ \}, \{\bar{\mathsf{s}}\}, \{\{\ \}, \{\ \}\}, T$. This allows us to distinguish between the effects before and after the opening of M. Figure 13 gives the typing rules for expressions.

$$\frac{\mathcal{U}, E \vdash M : Cap[F] \quad \mathcal{U}, E \vdash M' : Cap[F']}{\mathcal{U}, E \vdash M.M' : Cap[F \uplus F']} \text{(Exp . } \star)$$

$$\frac{\mathcal{U}, E \vdash M : \mathsf{s}_V^{\circ\{V', Z', \leftrightarrow\}}[T_U^Z] \quad \mathcal{U}, E \vdash T'}{\mathcal{U}, E \vdash \mathbf{in}\ M : Cap[\{\mathsf{s}\}, \{\ \}, T']} \text{(Exp } \mathbf{in}\ \star)$$

$$\frac{\mathcal{U}, E \vdash M : \mathsf{s}_V^{\circ\{V', Z', \leftrightarrow\}}[T_U^Z] \quad \mathcal{U}, E \vdash T'}{\mathcal{U}, E \vdash \overline{\mathbf{in}}\ M : Cap[\{\bar{\mathsf{s}}\}, \{\ \}, T']} \text{(Exp } \overline{\mathbf{in}}\ \star)$$

$$\frac{\mathcal{U}, E \vdash M : \mathsf{s}_V^{\circ\{V', Z', \leftrightarrow\}}[T_U^Z] \quad \mathcal{U}, E \vdash T'}{\mathcal{U}, E \vdash \mathbf{out}\ M : Cap[\{\mathsf{s}\}, \{\ \}, T']} \text{(Exp } \mathbf{out}\ \star)$$

$$\frac{\mathcal{U}, E \vdash M : \mathsf{s}_V^{\circ\{V', Z', \leftrightarrow\}}[T_U^Z] \quad \mathcal{U}, E \vdash T'}{\mathcal{U}, E \vdash \overline{\mathbf{out}}\ M : Cap[\{\bar{\mathsf{s}}\}, \{\ \}, T']} \text{(Exp } \overline{\mathbf{out}}\ \star)$$

$$\frac{\mathcal{U}, E \vdash M : \mathsf{s}_V^{\circ\{V', Z', U'\}}[T_U^Z] \quad \Im(\{V', Z', U'\}, \mathsf{s}) = \mathsf{S}, \mathsf{S}'}{\mathcal{U}, E \vdash \mathbf{open}\ M : Cap[\mathsf{S}, \{\mathsf{s}\} \cup \mathsf{S}', T]} \text{(Exp } \mathbf{open}\ \star)$$

$$\frac{\mathcal{U}, E \vdash M : \mathsf{s}_V^{\circ\{V', Z', U'\}}[T_U^Z]}{\mathcal{U}, E \vdash \overline{\mathbf{open}}\ M : Cap[\{\ \}, \{\bar{\mathsf{s}}\}, \{\{\ \}, \{\ \}\}, T]} \text{(Exp } \overline{\mathbf{open}}\ \star)$$

Fig. 13. Good expressions with expanded effects

We have also to define when an expanded effect agrees with an ambient type. An expanded effect $\mathsf{S}_1, \mathsf{S}_2, \{\mathsf{S}_3, \mathsf{S}_4\}, T$ agrees with an ambient type $\mathsf{s}_V^Y[T_U'^Z]$ iff:

- $T = T'$;
- $\mathsf{S}_1, \mathsf{S}_2$ agree with $\{V, Z, U\}$;
- $Y = \circ\{V', Z', U'\}$ and $\mathsf{S}_3, \mathsf{S}_4$ agree with $\{V', Z', U'\}$;

- all active security levels in $\mathsf{S}_1 \cup \mathsf{S}_3$ are $\leq^{\frown}$ s and all passive security levels in $\mathsf{S}_1 \cup \mathsf{S}_3$ are $\leq^{\frown}$ $\bar{\mathsf{s}}$;
- all active security levels in $\mathsf{S}_2 \cup \mathsf{S}_4$ are $\leq^{\circ}$ s and all passive security levels in $\mathsf{S}_2 \cup \mathsf{S}_4$ are $\leq^{\circ}$ $\bar{\mathsf{s}}$.

See Figure 14 for the formal definition.

Lastly, we need to combine the effects of two processes when they are put in parallel. If both effects are simple or expanded, we can use the $\uplus$ as defined in Figure 12 obtaining in both cases a simple effect. Otherwise we have that only one process contains an $\overline{\mathbf{open}}$ co-action, so we can obtain an expanded effect in which the sets of the simple effect are considered both before and after the opening action. This definition is given in Figure 15.

In the typing rule for processes we cannot combine expanded effects by means of set union as in rules (Proc Action) and (Proc Par). The rule (Proc Action $\star$) uses the operator $\uplus$ of Figure 12, since the action M will be done before the actions in P. Instead the typing rule (Proc Par $\star$) uses the operator $\overline{\uplus}$ of Figure 15, since the actions in P and in Q will be done in parallel. Lastly rule (Proc Input $\star$) takes into account that the effects of P can be either simple or expanded. The new rules are given in Figure 16. The remaining rules do not change, but we convene that all effects can be either simple or expanded.

$$\mathsf{S}_1, \mathsf{S}_2, \{\mathsf{S}_3, \mathsf{S}_4\}, T \bowtie \mathsf{s}_V^Y[T'^Z_U] \iff \begin{array}{l} T = T' \ \& \\ \mathsf{S}_1, \mathsf{S}_2 \bowtie \{V, U, Z\} \ \& \\ Y = \circ\{V', U', Z'\} \ \& \\ \mathsf{S}_3, \mathsf{S}_4 \bowtie \{V', U', Z'\} \ \& \\ \forall \mathsf{s}', \bar{\mathsf{s}}' \in \mathsf{S}_1 \cup \mathsf{S}_3 . \mathsf{s}' \leq^{\frown} \mathsf{s} \ \& \ \bar{\mathsf{s}}' \leq^{\frown} \bar{\mathsf{s}} \ \& \\ \forall \mathsf{s}', \bar{\mathsf{s}}' \in \mathsf{S}_2 \cup \mathsf{S}_4 . \mathsf{s}' \leq^{\circ} \mathsf{s} \ \& \ \bar{\mathsf{s}}' \leq^{\circ} \bar{\mathsf{s}} \end{array}$$

Fig. 14. Agreement between expanded effects and ambient types

$$\mathsf{S}, H, T \overline{\uplus} \mathsf{S}', H', T = \begin{cases} \mathsf{S} \cup \mathsf{S}', \mathsf{S}_1 \cup H', \{\mathsf{S}_2 \cup \mathsf{S}', \mathsf{S}_3 \cup H'\}, T & \text{if } H = \mathsf{S}_1, \{\mathsf{S}_2, \mathsf{S}_3\} \ \& \\ & \quad H' \subseteq \mathcal{U} \cup \overline{\mathcal{U}} \\ \mathsf{S} \cup \mathsf{S}', \mathsf{S}_1 \cup H, \{\mathsf{S}_2 \cup \mathsf{S}, \mathsf{S}_3 \cup H\}, T & \text{if } H \subseteq \mathcal{U} \cup \overline{\mathcal{U}} \ \& \\ & \quad H' = \mathsf{S}_1, \{\mathsf{S}_2, \mathsf{S}_3\} \\ \mathsf{S}, H, T \uplus \mathsf{S}', H', T' & \text{otherwise} \end{cases}$$

Fig. 15. Combination of simple and expanded effects for processes

Properties of Well Typed Processes. All the observations done for the system of Section 3 remain valid proviso that:

- if $\mathcal{U}, E \vdash P : \mathsf{S}_1, \mathsf{S}_2, \{\mathsf{S}_3 \cup \mathsf{S}_4\}, T$ we replace $\mathsf{S}_1 \cup \mathsf{S}_3$ to S and $\mathsf{S}_2 \cup \mathsf{S}_4$ to S';

$$\frac{\mathcal{U}, E, n_1 : W_1, \ldots, n_k : W_k \vdash P : \mathsf{S}, H, W_1 \times \ldots \times W_k}{\mathcal{U}, E \vdash (n_1 : W_1, \ldots, n_k : W_k).P : \mathsf{S}, H, W_1 \times \ldots \times W_k} \text{(Proc Input } \star\text{)}$$

$$\frac{\mathcal{U}, E \vdash M : Cap[F] \quad \mathcal{U}, E \vdash P : F'}{\mathcal{U}, E \vdash M.P : F \uplus F'} \text{(Proc Action } \star\text{)}$$

$$\frac{\mathcal{U}, E \vdash P : F \quad \mathcal{U}, E \vdash Q : F'}{\mathcal{U}, E \vdash P \mid Q : F \overline{\uplus} F'} \text{(Proc Par } \star\text{)}$$

Fig. 16. Good Processes with Expanded Effects

- we consider rule (Par Proc $\star$) instead of rule (Par Proc) in the discussion about single-threaded ambients.

Finally, to obtain a Soundness Result for the new type system we generalize the order relations to expanded effects as shown in Figure 17.

$$\begin{array}{lcl}
\mathsf{S}_3, \mathsf{S}_4, T \sqsubseteq \mathsf{S}'_1, \mathsf{S}'_2, \{\mathsf{S}'_3, \mathsf{S}'_4\}, T & \Longleftrightarrow & \mathsf{S}_3 \sqsubseteq^{\frown} \mathsf{S}'_3 \;\&\; \mathsf{S}_4 \sqsubseteq^{\circ} \mathsf{S}'_4 \\
\mathsf{S}_1, \mathsf{S}_2, \{\mathsf{S}_3, \mathsf{S}_4\}, T \sqsubseteq \mathsf{S}'_1, \mathsf{S}'_2, \{\mathsf{S}'_3, \mathsf{S}'_4\}, T & \Longleftrightarrow & \mathsf{S}_i \sqsubseteq^{\frown} \mathsf{S}'_i \text{ for } i = 1, 3 \\
 & & \;\&\; \mathsf{S}_i \sqsubseteq^{\circ} \mathsf{S}'_i \text{ for } i = 2, 4
\end{array}$$

Fig. 17. Partial order of simple and expanded effects

Theorem 2. *If $\mathcal{U}, E \vdash P : F$ and $P \longrightarrow Q$ then $\mathcal{U}, E \vdash Q : F'$ for some F' such that $F' \sqsubseteq F$.*

Proof. The proof is by induction on the definition of $\longrightarrow$.

We consider only rules (R-in) and (R-open). The case of (R-out) is similar to that of (R-in) and the other rules follow easily by induction.

For rule (R-in) the key observation is that

$$\text{if } F \bowtie \mathsf{s}_V^Y[T_U^Z] \text{ and } F' \sqsubseteq F \text{ then } F' \bowtie \mathsf{s}_V^Y[T_U^Z].$$

This follows easily from the definitions of $\bowtie$ (Figure 8) and $\sqsubseteq$ (Figure 10). The typing of the left-hand side contains the sub-derivations shown in Figure 18. The two applications of rule (Proc Amb) require respectively $((\{\mathsf{s}\}, \{\ \}, T') \uplus F_1) \overline{\uplus} F_2 \bowtie \mathsf{s}'^{Y'}_{V'}[T'^{Z'}_{U'}]$ and $((\{\bar{\mathsf{s}}\}, \{\ \}, T) \uplus F'_1) \overline{\uplus} F'_2 \bowtie \mathsf{s}_V^Y[T_U^Z]$. Therefore we can deduce the same type for the right-hand side as shown in Figure 19, where the two applications of rule (Proc Amb) require respectively $F_1 \overline{\uplus} F_2 \bowtie \mathsf{s}'^{Y'}_{V'}[T'^{Z'}_{U'}]$ and $F'_1 \overline{\uplus} F'_2 \bowtie \mathsf{s}_V^Y[T_U^Z]$.

For rule (R-open) notice that:

$$\text{if } \mathsf{S}_1, \mathsf{S}_2, T \bowtie \mathsf{s}_V^{\circ}[T_U^Z] \text{ then } \mathsf{S}_1, \mathsf{S}_2, T \sqsubseteq \Im(\{V, Z, U\}, \mathsf{s}), T;$$
$$\text{if } \mathsf{S}_1, \mathsf{S}_2, \{\mathsf{S}_3, \mathsf{S}_4\}, T \bowtie \mathsf{s}_V^{\circ\{V', Z', U'\}}[T_U^Z] \text{ then } \mathsf{S}_3, \mathsf{S}_4, T \sqsubseteq \Im(\{V', Z', U'\}, \mathsf{s}), T.$$

$$
\dfrac{\dfrac{\dfrac{\mathcal{U}, E \vdash m : \mathsf{s}_V^Y[T_U^Z] \quad \mathcal{U}, E \vdash T'}{\mathcal{U}, E \vdash \mathbf{in}\ m : Cap[\{\mathsf{s}\}, \{\ \}, T']} \quad \mathcal{U}, E \vdash P_1 : F_1}{\mathcal{U}, E \vdash \mathbf{in}\ m.P_1 : (\{\mathsf{s}\}, \{\ \}, T') \uplus F_1} \qquad \mathcal{U}, E \vdash P_2 : F_2}{\mathcal{U}, E \vdash \mathbf{in}\ m.P_1 \mid P_2 : ((\{\mathsf{s}\}, \{\ \}, T') \uplus F_1) \overline{\uplus} F_2}
$$

$$
\dfrac{\mathcal{U}, E \vdash n : \mathsf{s}'^{Y'}_{V'}[T'^{Z'}_{U'}] \quad \mathcal{U}, E \vdash \mathbf{in}\ m.P_1 \mid P_2 : ((\{\mathsf{s}\}, \{\ \}, T') \uplus F_1) \overline{\uplus} F_2 \quad \mathcal{U}, E \vdash T''}{\mathcal{U}, E \vdash n[\mathbf{in}\ m.P_1 \mid P_2] : \{\ \}, \{\ \}, T''}
$$

$$
\dfrac{\dfrac{\dfrac{\mathcal{U}, E \vdash m : \mathsf{s}_V^Y[T_U^Z] \quad \mathcal{U}, E \vdash T}{\mathcal{U}, E \vdash \overline{\mathbf{in}}\ m : Cap[\{\overline{\mathsf{s}}\}, \{\ \}, T]} \quad \mathcal{U}, E \vdash Q_1 : F_1'}{\mathcal{U}, E \vdash \overline{\mathbf{in}}\ m.Q_1 : (\{\overline{\mathsf{s}}\}, \{\ \}, T) \uplus F_1'} \qquad \mathcal{U}, E \vdash Q_2 : F_2'}{\mathcal{U}, E \vdash \overline{\mathbf{in}}\ m.Q_1 \mid Q_2 : ((\{\overline{\mathsf{s}}\}, \{\ \}, T) \uplus F_1') \overline{\uplus} F_2'}
$$

$$
\dfrac{\mathcal{U}, E \vdash m : \mathsf{s}_V^Y[T_U^Z] \quad \mathcal{U}, E \vdash \overline{\mathbf{in}}\ m.Q_1 \mid Q_2 : ((\{\overline{\mathsf{s}}\}, \{\ \}, T) \uplus F_1') \overline{\uplus} F_2' \quad \mathcal{U}, E \vdash T''}{\mathcal{U}, E \vdash m[\overline{\mathbf{in}}\ m.Q_1 \mid Q_2] : \{\ \}, \{\ \}, T''}
$$

Fig. 18. Typing of the left-hand side of rule (R-in)

$$
\dfrac{\mathcal{U}, E \vdash n : \mathsf{s}'^{Y'}_{V'}[T'^{Z'}_{U'}] \quad \dfrac{\mathcal{U}, E \vdash P_1 : F_1 \quad \mathcal{U}, E \vdash P_2 : F_2}{\mathcal{U}, E \vdash P_1 \mid P_2 : F_1 \overline{\uplus} F_2} \quad \mathcal{U}, E \vdash T}{\mathcal{U}, E \vdash n[P_1 \mid P_2] : \{\ \}, \{\ \}, T}
$$

$$
\dfrac{\mathcal{U}, E \vdash n[P_1 \mid P_2] : \{\ \}, \{\ \}, T \quad \dfrac{\mathcal{U}, E \vdash Q_1 : F_1' \quad \mathcal{U}, E \vdash Q_2 : F_2'}{\mathcal{U}, E \vdash Q_1 \mid Q_2 : F_1' \overline{\uplus} F_2'}}{\mathcal{U}, E \vdash n[P_1 \mid P_2] \mid Q_1 \mid Q_2 : F_1' \overline{\uplus} F_2'}
$$

$$
\dfrac{\mathcal{U}, E \vdash m : \mathsf{s}_V^Y[T_U^Z] \quad \mathcal{U}, E \vdash n[P_1 \mid P_2] \mid Q_1 \mid Q_2 : F_1' \overline{\uplus} F_2' \quad \mathcal{U}, E \vdash T''}{\mathcal{U}, E \vdash m[n[P_1 \mid P_2] \mid Q_1 \mid Q_2] : \{\ \}, \{\ \}, T''}
$$

Fig. 19. Typing of the right-hand side of rule (R-in)

This is a consequence of the definitions of $\bowtie$ (Figure 8), $\sqsubseteq$ (Figure 10), and $\Im(\ ,\ )$ (Figure 6). Figure 20 shows the typing of the left-hand side, under the assumption that $Y = \circ\{V', Z', U'\}$ and $\Im(\{V', Z', U'\}, \mathsf{s}) = \mathsf{S}, \mathsf{S}'$ (the case $Y = \circ$ and $\Im(\{V, Z, U\}, \mathsf{s}) = \mathsf{S}, \mathsf{S}'$ is similar and simpler). The application of rule (Proc Amb) requires $((\{\ \}, \{\bar{\mathsf{s}}\}, \{\{\ \}, \{\ \}\}, T) \uplus F_1) \overline{\uplus} F_2 \bowtie \mathsf{s}_V^Y[T_U^Z]$, therefore $((\{\ \}, \{\bar{\mathsf{s}}\}, \{\{\ \}, \{\ \}\}, T) \uplus F_1) \overline{\uplus} F_2$ must be an expanded effect. Let

$$((\{\ \}, \{\bar{\mathsf{s}}\}, \{\{\ \}, \{\ \}\}, T) \uplus F_1) \overline{\uplus} F_2 = \mathsf{S}_1, \mathsf{S}_2, \{\mathsf{S}_3, \mathsf{S}_4\}, T.$$

Then by the definitions of $\uplus$ (Figure 12), of $\overline{\uplus}$ (Figure 15), and of $\sqsubseteq$ we get $F_1 \overline{\uplus} F_2 \sqsubseteq \mathsf{S}_3, \mathsf{S}_4, T$. From the above remark $\mathsf{S}_1, \mathsf{S}_2, \{\mathsf{S}_3, \mathsf{S}_4\}, T \bowtie \mathsf{s}_V^Y[T_U^Z]$ implies $\mathsf{S}_3, \mathsf{S}_4, T \sqsubseteq \mathsf{S}, \mathsf{S}', T$. We obtain $F_1 \overline{\uplus} F_2 \sqsubseteq \mathsf{S}, \mathsf{S}', T$ and also $F \overline{\uplus} (F_1 \overline{\uplus} F_2) \sqsubseteq (\mathsf{S}, \{\mathsf{s}\} \cup \mathsf{S}', T) \overline{\uplus} F$ being $\overline{\uplus}$ commutative. We are done, since we can derive $F \overline{\uplus} (F_1 \overline{\uplus} F_2)$ for the right-hand side as shown in Figure 21.

$$\dfrac{\dfrac{\mathcal{U}, E \vdash n : \mathsf{s}_V^Y[T_U^Z] \quad \mathcal{U}, E \vdash T}{\mathcal{U}, E \vdash \mathbf{open}\ n : \mathit{Cap}[\mathsf{S}, \{\mathsf{s}\} \cup \mathsf{S}', T]} \quad \mathcal{U}, E \vdash P : F}{\mathcal{U}, E \vdash \mathbf{open}\ n.P : (\mathsf{S}, \{\mathsf{s}\} \cup \mathsf{S}', T) \uplus F}$$

$$\dfrac{\dfrac{\dfrac{\mathcal{U}, E \vdash n : \mathsf{s}_V^Y[T_U^Z] \qquad \mathcal{U}, E \vdash T}{\mathcal{U}, E \vdash \overline{\mathbf{open}}\ n : \mathit{Cap}[\{\ \}, \{\bar{\mathsf{s}}\}, \{\{\ \}, \{\ \}\}, T]} \quad \mathcal{U}, E \vdash Q_1 : F_1}{\mathcal{U}, E \vdash \overline{\mathbf{open}}\ n.Q_1 : (\{\ \}, \{\bar{\mathsf{s}}\}, \{\{\ \}, \{\ \}\}, T) \uplus F_1} \qquad \mathcal{U}, E \vdash Q_2 : F_2}{\mathcal{U}, E \vdash \overline{\mathbf{open}}\ n.Q_1 \mid Q_2 : ((\{\ \}, \{\bar{\mathsf{s}}\}, \{\{\ \}, \{\ \}\}, T) \uplus F_1) \overline{\uplus} F_2}$$

$$\dfrac{\mathcal{U}, E \vdash n : \mathsf{s}_V^Y[T_U^Z] \quad \mathcal{U}, E \vdash \overline{\mathbf{open}}\ n.Q_1 \mid Q_2 : ((\{\ \}, \{\bar{\mathsf{s}}\}, \{\{\ \}, \{\ \}\}, T) \uplus F_1) \overline{\uplus} F_2 \quad \mathcal{U}, E \vdash T}{\mathcal{U}, E \vdash n[\overline{\mathbf{open}}\ n.Q_1 \mid Q_2] : \{\ \}, \{\ \}, T}$$

$$\dfrac{\mathcal{U}, E \vdash \mathbf{open}\ n.P : (\mathsf{S}, \{\mathsf{s}\} \cup \mathsf{S}', T) \uplus F \quad \mathcal{U}, E \vdash n[\overline{\mathbf{open}}\ n.Q_1 \mid Q_2] : \{\ \}, \{\ \}, T}{\mathcal{U}, E \vdash \mathbf{open}\ n.P \mid n[\overline{\mathbf{open}}\ n.Q_1 \mid Q_2] : (\mathsf{S}, \{\mathsf{s}\} \cup \mathsf{S}', T) \uplus F}$$

Fig. 20. Typing of the left-hand side of rule (R-open)

5 Examples

In this section we present some examples, where we assume a different security level for each ambient. Examples in Subsection 5.2, 5.3 and 5.4 are discussed both in [9] and [2]: we present them in order to show which kinds of typing

$$\dfrac{\mathcal{U}, E \vdash P : F \qquad \dfrac{\mathcal{U}, E \vdash Q_1 : F_1 \quad \mathcal{U}, E \vdash Q_2 : F_2}{\mathcal{U}, E \vdash Q_1 \mid Q_2 : F_1 \boxplus F_2}}{\mathcal{U}, E \vdash P \mid Q_1 \mid Q_2 : F \boxplus (F_1 \boxplus F_2)}$$

Fig. 21. Typing of the right-hand side of rule (R-open)

are derivable in our type system and which kinds of constraints on the security levels order are required to type them. The other two examples, instead, show possible applications of our type system, in particular how to use it to ensure some security properties. Examples 5.2, 5.4 and 5.5 can be typed in the simpler system of Section 3 while the others examples require the extension discussed in Section 4.

5.1 Mailserver

In this example we model a mailserver. In our model, a mailserver MS is an ambient which contains a set of mailboxes, $MBOX_i$, for $1 \leq i \leq k$. A message MSG is a mobile ambient that enters first in the mailserver and then in a mailbox. Messages have high level mobility rights, whereas each user must have the same mobility rights of its mailbox. The idea is that users do not have rights to enter mailboxes of other users. A user U_i that tries to use a capability $\mathbf{in}\ mb_j$, for $i \neq j$ is not a well-typed process.

$$\begin{aligned}
MSG &= m[\mathbf{in}\ ms.\mathbf{in}\ mb_i.\overline{\mathbf{open}}\ m.\langle M \rangle] \\
MS &= ms[!\overline{\mathbf{in}}\ ms \mid !\overline{\mathbf{out}}\ ms \mid MBOX_1 \mid \ldots \mid MBOX_k] \\
MBOX_i &= mb_i[!\overline{\mathbf{in}}\ mb_i \mid !\overline{\mathbf{out}}\ mb_i] \\
U_i &= u_i[\mathbf{in}\ ms.\mathbf{in}\ mbox_i.\mathbf{open}\ m.(x : T).\mathbf{out}\ mb_i.\mathbf{out}\ ms.P]
\end{aligned}$$

where T is the exchange type of P.

Considering an environment E such that:

$$\begin{array}{l}
\mathcal{U}, E \vdash m : \mathsf{msg}_{\frac{\circ}{\bullet}}^{\{\overline{\bullet}, \underline{\vee}, \oplus\}}[T_{\oplus}^{\curvearrowright}] \\
\mathcal{U}, E \vdash ms : \mathsf{mserv}_{\frac{\bullet}{\bullet}}[T_{\frac{\vee}{\leftrightarrow}}] \\
\mathcal{U}, E \vdash mb_i : \mathsf{mbox}_{i\frac{\bullet}{\bullet}}[T_{\frac{\vee}{\leftrightarrow}}] \\
\mathcal{U}, E \vdash u_i : \mathsf{ut}_{i\frac{\bullet}{\circ}}[T_{\oplus}^{\curvearrowright}] \\
\mathcal{U}, E \vdash M : T
\end{array}$$

we can type processes inside ambients as follows:

$$\begin{array}{l}
\mathcal{U}, E \vdash \mathbf{in}\ ms.\mathbf{in}\ mb_i.\overline{\mathbf{open}}\ m.\langle M \rangle : \{\mathsf{mserv}, \mathsf{mbox}_i\}, \{\overline{\mathsf{msg}}\}, \{\varnothing, \varnothing\}, T \\
\mathcal{U}, E \vdash !\overline{\mathbf{in}}\ ms \mid !\overline{\mathbf{out}}\ ms \mid MBOX_1 \mid \ldots \mid MBOX_k : \{\overline{\mathsf{mserv}}\}, \{\}, T \\
\mathcal{U}, E \vdash !\overline{\mathbf{in}}\ mb_i \mid !\overline{\mathbf{out}}\ mb_i : \{\overline{\mathsf{mbox}_i}\}, \{\}, T \\
\mathcal{U}, E \vdash \mathbf{in}\ ms.\mathbf{in}\ mb_i.\mathbf{open}\ m.(x : T).\mathbf{out}\ mb_i.\mathbf{out}\ ms.P : \{\mathsf{mserv}, \mathsf{mbox}_i\}, \{\mathsf{msg}\}, T
\end{array}$$

It easy to check that process types agree with corresponding ambient types. In order to avoid unauthorized access to mailboxes, we consider the following order defined on $\mathcal{U}$:

$$\mathsf{msg} \geq^{\frown} \mathsf{ut}_i =^{\frown} \mathsf{mbox}_i \qquad \text{for all } i$$

$$\mathsf{ut}_i \not\geq^{\frown} \mathsf{ut}_j \;\&\; \mathsf{ut}_j \not\geq^{\frown} \mathsf{ut}_i \qquad \text{for all } i \neq j$$

and

$$\mathsf{ut}_i \geq^{\circ} \mathsf{msg} \qquad \text{for all } i.$$

5.2 Renaming

We recall the construct given in [9] to change the name of an ambient:

$$n\ \mathbf{be}\ m.P = m[\mathbf{out}\ n.\overline{\mathbf{in}}\ m.\mathbf{open}\ n.P] \mid \overline{\mathbf{out}}\ n.\mathbf{in}\ m.\overline{\mathbf{open}}\ n$$

Let E be an environment such that:

$$\begin{array}{l} \mathcal{U}, E \vdash m : \mathsf{sm}^{Y}_{\circ}[T^{\frown}_{U}] \\ \mathcal{U}, E \vdash n : \mathsf{sn}^{\circ}_{\circ}[T^{\frown}_{\leftrightarrow}] \end{array}$$

where Y, U and T depend on the type of P. We assume that in the environment E we can derive the following typing judgments for P and Q:

$$\begin{array}{l} \mathcal{U}, E \vdash P : \mathsf{S}^{P}_{1}, \mathsf{S}^{P}_{2}, T \\ \mathcal{U}, E \vdash Q : \mathsf{S}^{Q}_{1}, \mathsf{S}^{Q}_{2}, T \end{array}$$

The desired property is that[1]:

$$n[n\ \mathbf{be}\ m.P \mid Q] \simeq m[P \mid Q]$$

for all Q such that no ambient in Q can perform an **out** n action. In our type system, this last condition can be assumed by requiring that $\mathsf{sn} \notin \mathsf{S}^{Q}_{1}$.

In such an environment we can derive the following typing for processes:

$$\begin{array}{l} \mathcal{U}, E \vdash \mathbf{out}\ n.\overline{\mathbf{in}}\ m.\mathbf{open}\ n.P : \\ \qquad\qquad (\{\mathsf{sn}, \overline{\mathsf{sm}}, \overline{\mathsf{sn}}\}, \{\mathsf{sn}, \overline{\mathsf{sn}}\}, T) \uplus (\mathsf{S}^{P}_{1}, \mathsf{S}^{P}_{2}, T) \\ \mathcal{U}, E \vdash \overline{\mathbf{out}}\ n.\mathbf{in}\ m.\overline{\mathbf{open}}\ n : \{\overline{\mathsf{sn}}, \mathsf{sm}\}, \{\overline{\mathsf{sn}}\}, T \end{array}$$

Finally we observe that we can type the renaming protocol under the following conditions on the security levels order:

$$\begin{array}{l} \mathsf{sm} \geq^{\circ} \mathsf{sn} \\ \mathsf{sm} =^{\frown} \mathsf{sn} \end{array}$$

In particular, it seems meaningful that security levels sm and sn are the same.

5.3 Firewall

The untyped protocol given in [9] for controlling access through a firewall is:

$$\begin{array}{rl} AG = & m[\overline{\mathbf{in}}\ m.\mathbf{open}\ k. \\ & \qquad (x)x.\overline{\mathbf{open}}\ m.Q] \\ FW = & (\nu w)(w[\overline{\mathbf{out}}\ w.\overline{\mathbf{in}}\ w.\mathbf{open}\ m.P \\ & \qquad \mid k[\mathbf{out}\ w.\mathbf{in}\ m.\overline{\mathbf{open}}\ k.\langle \mathbf{in}\ w\rangle]\ ]) \end{array}$$

FW represents the firewall and AG a trusted agent which crosses the firewall thanks to the pilot ambient k. The name k plays the role of a key which allows only authorized agents to cross the firewall.

[1] As in [9] two processes are $\simeq$-equivalent iff no closing contexts can tell them apart.

Let P and Q be such that:

$$\begin{array}{l} \mathcal{U}, E \vdash Q : \{\ \}, \mathsf{S}_2^Q, T \\ \mathcal{U}, E \vdash P : \{\ \}, \mathsf{S}_2^Q, T \end{array}$$

where $T = Cap[\{fw\}, \{\ \}, T']$ for some T' and we assume that processes P and Q do not perform move actions to show that in this case we can type w as immobile. Let E be an environment such that:

$$\begin{array}{l} \mathcal{U}, E \vdash m : \mathsf{ag}^{\circ\{\underline{\vee}, \oplus, \overline{\bullet}\}}_{\overline{\circ}}[T^{\curvearrowright}_{\leftrightarrow}] \\ \mathcal{U}, E \vdash w : \mathsf{fw}^{\bullet}_{\overline{\circ}}[T^{\vee}_{\overline{\leftrightarrow}}] \\ \mathcal{U}, E \vdash k : \mathsf{aut}^{\circ\{\curvearrowright, \oplus, \overline{\bullet}\}}_{\overline{\circ}}[T^{\curvearrowright}_{\oplus}] \end{array}$$

Let P_m (resp. P_w and P_k) be the process inside the ambient m (resp. w and k). In the environment E, we can derive the following judgments:

$$\begin{array}{l} \mathcal{U}, E \vdash P_m : \{\mathsf{fw}, \overline{\mathsf{ag}}\}, \{\mathsf{aut}, \overline{\mathsf{ag}}, \{\{\ \}, \mathsf{S}_2^Q\}\}, T \\ \mathcal{U}, E \vdash P_w : \{\overline{\mathsf{fw}}\}, \{\mathsf{ag}\}, T \\ \mathcal{U}, E \vdash P_k : \{\mathsf{fw}, \mathsf{ag}\}, \{\overline{\mathsf{aut}}\}, T \end{array}$$

and we obtain the following typed versions of the protocols:

$$\begin{array}{rl} AG = & m[\overline{\mathbf{in}}\ m.\mathbf{open}\ k. \\ & \quad (x : Cap[\{\mathsf{fw}\}, \{\ \}, T'])x.\overline{\mathbf{open}}\ m.Q] \\ FW = & (\nu w : \mathsf{fw}^{\bullet}_{\overline{\circ}}[T^{\vee}_{\overline{\leftrightarrow}}])(w[\overline{\mathbf{out}}\ w.\overline{\mathbf{in}}\ w.\mathbf{open}\ m.P \\ & \quad \mid k[\mathbf{out}\ w.\mathbf{in}\ m.\overline{\mathbf{open}}\ k.\langle \mathbf{in}\ w\rangle]\ ]) \end{array}$$

We can type the ambient w as immobile, since opening the mobile ambient m does not unleash mobility capabilities. We obtain this information from the type of m and the possibility to type P_m with an expanded effect. The presence of the co-action $\overline{\mathbf{open}}\ m$ is crucial for having such control over the behaviour of an ambient.

The demanded property is that:

$$(\nu m)(AG \mid FW) \simeq (\nu m, w)(w[P \mid Q])$$

under the conditions $w \notin fn(Q)$ and $m \notin fn(P)$. These conditions are assured by $\mathsf{fw} \notin \mathsf{S}_2^Q$ and $\mathsf{ag} \notin \mathsf{S}_2^P$. We do not see how to express this requirement using only types in the system of [9].

Finally, we observe that the following constraints on the order relation among security levels are needed to correctly type the whole protocol:

$$\mathsf{fw} \leq^{\curvearrowright} \mathsf{ag} \leq^{\curvearrowright} \mathsf{aut}$$

and

$$\mathsf{fw} \geq^{\circ} \mathsf{ag} \geq^{\circ} \mathsf{aut}$$

5.4 Channels

The encoding of asynchronous typed π-calculus is an important test of expressiveness for the ambient calculus. Several encodings have been proposed for both typed and untyped mobile ambients and for safe ambients. Here we consider the typed version in our system of the simpler untyped protocol presented in [9]:

$$
\begin{aligned}
[\![p(q_1,\ldots,q_n)]\!] &= p[\overline{\mathbf{in}}\, p.\mathbf{open}\, p.(q_1,\ldots,q_n).\overline{\mathbf{open}}\, p.[\![P]\!] \mid \mathbf{open}\, p \\
[\![\overline{p}\langle q_1,\ldots,q_n\rangle]\!] &= p[\mathbf{in}\, p.\overline{\mathbf{open}}\, p.\langle q_1,\ldots,q_n\rangle] \\
[\![(\nu p:T)P]\!] &= (\nu p:[\![T]\!])[\![P]\!] \\
[\![P \mid Q]\!] &= [\![P]\!] \mid [\![Q]\!] \\
[\![!P]\!] &= ![\![P]\!] \\
[\![\mathbf{0}]\!] &= \mathbf{0} \\
[\![Ch(T_1,\ldots,T_n)]\!] &= \mathsf{Proc}^{\circ}_{\circ}[[\![T_1]\!] \times \ldots \times [\![T_n]\!]^{\curvearrowright}_{\leftrightarrow}] \\
[\![p:T]\!] &= p:[\![T]\!] \\
[\![\Gamma, p:T]\!] &= [\![\Gamma]\!], [\![p:T]\!]
\end{aligned}
$$

where the type grammar is $T := Ch(T_1,\ldots,T_n)$, $n \geq 0$.

If $\Gamma \vdash P$ can be derived using the standard rules for the asynchronous typed π-calculus [9] then we get either $[\![\Gamma]\!] \vdash [\![P]\!] : \{\ \}, \{\mathsf{Proc}\}, Shh$ or $[\![\Gamma]\!] \vdash [\![P]\!] : \{\ \}, \{\}, Shh$.

5.5 Secure Channels

This example is inspired by the information flow analysis carried out in [7]. We want to model a π-calculus, in which to each channel p is associated a write and a read access right. Moreover, a π-calculus process can be defined as running at a given security level[2]. According with the principle that information can move only upward, the security policy imposes that a channel can communicate information only to π-calculus processes with a security level greater than its write access right and can receive information only from π-calculus processes with a security level less than its read access right. In the π-calculus process:

$$\sigma\{p\langle q\rangle\} \mid \rho\{p(q)P\}$$

we have that the process $\sigma\{p\langle q\rangle\}$ runs at the security level σ and the process $\rho\{p(q)P\}$ runs at the security level ρ. Communication can take place if the write access rights of p, W_p, are less or equal to ρ and if the read access rights of p, R_p, are greater or equal to σ.

We show an encoding of π-calculus channels with read and write access rights into Safe Ambients. The basic idea is to modify the above encoding of channels in Subsection 5.4 in such a way:

- the order $\leq^{\circ}$ among security levels in ambient types represents the order among π-calculus security levels;
- each channel p is represented as a pair of ambients p^w and p^r; such ambients belong to security levels w_p and r_p respectively;
- the security levels w_p and r_p play in the ambient encoding of secure channels the role of access rights W_p and R_p;
- in the encoding of a π-calculus process $\rho\{P\}$, an ambient ρ, belonging to a security level s_ρ, represents the security level of a process;

[2] "security level" is overloaded in this subsection, since we consider both security levels of π-calculus processes and security level of ambient types. No ambiguity arises thanks to the context.

– in the encoding of the π-calculus process $\sigma\{p\langle q\rangle\} \mid \rho\{p(q)P\}$, the communication protocol imposes that the ambient ρ must be able to open p^w, and p^r must be able to open σ, so that the protocol is well-typed only if:

$$
\begin{aligned}
&\mathsf{s}_\rho \geq^\circ \mathsf{w}_p \\
&\mathsf{s}_\sigma \leq^\circ \mathsf{r}_p
\end{aligned}
$$

The encoding is rather complicated, because a process does not know the security level of the process with which communication will happen and hence we need two auxiliary ambients p^m and p^n. This implies that also the channel q will be represented by four ambients, q^r, q^w, q^m, q^n.

$$
\begin{aligned}
[\![\, \sigma\{p\langle q\rangle\}]\!] = \sigma[&\mathbf{in}\; p^r.\overline{\mathbf{out}}\; \sigma.\overline{\mathbf{open}}\; \sigma \\
&\mid p^m[\mathbf{out}\; \sigma.\overline{\mathbf{open}}\; p^m.\mathbf{open}\; \sigma] \\
&\mid p^w[\overline{\mathbf{in}}\; p^w.\mathbf{open}\; p^n.\overline{\mathbf{open}}\; p^w.\langle q^r, q^w, q^m, q^n\rangle] \\
&]
\end{aligned}
$$

$$
\begin{aligned}
[\![\, \rho\{p(x)P\}]\!] = p^r[&\overline{\mathbf{in}}\; p^r.\mathbf{open}\; p^m.\overline{\mathbf{out}}\; p^r.\mathbf{in}\; \rho.\overline{\mathbf{open}}\; p^r.(x^r, x^w, x^m, x^n).[\![\, P]\!] \\
&\mid p^n[\mathbf{in}\; p^w.\overline{\mathbf{open}}\; p^n.\mathbf{in}\; \rho] \\
&\mid \rho[\overline{\mathbf{in}}\; \rho.\mathbf{open}\; p^w.\mathbf{out}\; p^r.\overline{\mathbf{in}}\; \rho.\overline{\mathbf{open}}\; \rho] \\
&] \\
&\mid \mathbf{open}\; p^r \mid \mathbf{open}\; \rho
\end{aligned}
$$

We have that:

$$
[\![\, \rho\{p(x)P\}]\!] \mid [\![\, \sigma\{p\langle q\rangle\}]\!] \longrightarrow [\![\, P]\!][\{x^r := q^r, x^w := q^w, x^m := q^m, x^n := q^n\}
$$

Let $[\![\, Ch(T)]\!] = \mathsf{w}\frac{\circ}{\circ}[[\![\, T]\!]^{\curvearrowright}_{\leftrightarrow}] \times \mathsf{r}\frac{\circ}{\circ}[[\![\, T]\!]^{\curvearrowright}_{\leftrightarrow}] \times \mathsf{w}\frac{\circ}{\circ}[[\![\, T]\!]^{\curvearrowright}_{\leftrightarrow}] \times \mathsf{r}\frac{\circ}{\circ}[[\![\, T]\!]^{\curvearrowright}_{\leftrightarrow}]$, where w (resp. r) is the write (resp. read) access right of a channel of type T and suppose the channel q has type T in the typed π-calculus. We can type the above protocol in an environment E such that:

$$
\begin{aligned}
&\mathcal{U}, E \vdash \sigma : \mathsf{s}_\sigma\tfrac{\circ}{\bullet}[[\![\, Ch(T)]\!]^{\curvearrowright}_{\leftrightarrow}] \\
&\mathcal{U}, E \vdash \rho : \mathsf{s}_\rho\tfrac{\circ}{\circ}[[\![\, Ch(T)]\!]^{\curvearrowright}_{\leftrightarrow}] \\
&\mathcal{U}, E \vdash p^w, p^n : \mathsf{w}_p\tfrac{\circ}{\circ}[[\![\, Ch(T)]\!]^{\curvearrowright}_{\leftrightarrow}] \\
&\mathcal{U}, E \vdash p^r, p^m : \mathsf{r}_p\tfrac{\circ}{\circ}[[\![\, Ch(T)]\!]^{\curvearrowright}_{\leftrightarrow}] \\
&\mathcal{U}, E \vdash q^w, q^n : \mathsf{w}_q\tfrac{\circ}{\circ}[[\![\, Ch(T)]\!]^{\curvearrowright}_{\leftrightarrow}] \\
&\mathcal{U}, E \vdash q^r, q^m : \mathsf{r}_q\tfrac{\circ}{\circ}[[\![\, Ch(T)]\!]^{\curvearrowright}_{\leftrightarrow}]
\end{aligned}
$$

6 Conclusion

We presented two type systems for safe ambients that guarantee security properties. While the first one could be adapted without problems to mobile ambients, the second one requires co-actions in order to distinguish the effects before and after a (unique!) $\overline{\mathbf{open}}$ action.

Both type systems describe in a rather precise way the behaviours of ambients and could be used to establish a typed equivalence theory of ambients, similarly to what was done for the π–calculus in [11].

The typing rules are syntax directed and therefore there is a type checking algorithm quite similar to that of [12]. It is not clear to us what means type inference for systems which look for security properties, since these are just expressed by the environments and by the types of bound names.

Acknowledgment. The authors are grateful to the referees for their helpful comments.

References

1. Michele Bugliesi and Giuseppe Castagna. Secure safe ambients. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages*, 2001. To appear.
2. Luca Cardelli, Giorgio Ghelli, and Andrew D. Gordon. Mobility types for mobile ambients. In Jiří Wiederman, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP 1999*, volume 1644 of *Lecture Notes in Computer Science*, pages 230–239. Springer-Verlag, 1999.
3. Luca Cardelli, Giorgio Ghelli, and Andrew D. Gordon. Ambient groups and mobility types. In Jan van Leeuwen et al., editor, *Theoretical Computer Science: Exploring New Frontiers in Theoretical Informatics*, volume 1872 of *Lecture Notes in Computer Science*, pages 333–347. Springer-Verlag, 2000.
4. Luca Cardelli and Andrew D. Gordon. Mobile ambients. In Maurice Nivat, editor, *FoSSaCS 1998*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer-Verlag, 1998.
5. Luca Cardelli and Andrew D. Gordon. Types for mobile ambients. In *POPL'99*, pages 79–92, New York, NY, USA, 1999. ACM Press.
6. Matthew Hennessy and James Riely. Resource access control in systems of mobile agents (extended abstract). In *Proc. of 3rd International Workshop on High-Level Concurrent Languages (HLCL'98)*, 1998. Vol. 16(3) of *Electronic Notes in Theoretical Computer Science*. Full version to appear in *Information and Computation*.
7. Matthew Hennessy and James Riely. Information flow vs. resource access in the asynchronous π-calculus. In Ugo Montanari, José Rolim, and Emo Welzl, editors, *ICALP 2000*, volume 1853 of *Lecture Notes in Computer Science*, pages 415–427. Springer-Verlag, 2000.
8. Francesca Levi and Chiara Bodei. Security analysis for mobile ambients. In *Proceedings of the Workshop on Issues in the Theory of Security,* (co-located with ICALP 2000), pages 18–23, 2000.
9. Francesca Levi and Davide Sangiorgi. Controlling interference in ambients. In *POPL'00*, pages 352–364, New York, NY, USA, 2000. ACM Press.
10. Rocco De Nicola, Gianluigi Ferrari, Rosario Pugliese, and Betty Venneri. Types for access control. *Theoretical Computer Science*, 2000. Special issue on Coordination. To appear, available at `http://rap.dsi.unifi.it/papers.htm`.
11. Benjamin Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. In *Logic in Computer Science*, 1993. Full version in *Mathematical Structures in Computer Science*, Vol. 6, No. 5, pages 409-454, 1996.
12. Pascal Zimmer. Subtyping and typing algorithms for mobile ambients. In Jerzy Tiuryn, editor, *FoSSaCS 2000*, volume 1784 of *Lecture Notes in Computer Science*, pages 375–390. Springer-Verlag, 2000.