Access control for mobile agents: the calculus of Boxed Ambients

MICHELE BUGLIESI
Università 'Ca' Foscari', Venezia, Italy
and
GIUSEPPE CASTAGNA
École Normale Supérieure, Paris, France
and
SILVIA CRAFA
Università 'Ca' Foscari', Venezia, Italy
École Normale Supérieure, Paris, France

Boxed Ambients are a variant of Mobile Ambients that result from dropping the `open` capability, and introducing new primitives for ambient communication. The new model of communication is faithful to the principles of distribution and location-awareness of Mobile Ambients, and complements the constructs `in` and `out` for mobility with finer-grained mechanisms for ambient interaction. We introduce the new calculus, study the impact of the new mechanisms for communication o typing and mobility, and show that they yield an effective framework for resource protection and access control in distributed systems.

Categories and Subject Descriptors: F.3.1 [**Logics and Meaning of Programs**]: Specifying and Verifying and Reasoning about Programs—*Logics of Programs*; F.3.3 [**Studies of Program Constructs**]: Type structure; D.1.3 [**Programming Techniques**]: Concurrent Programming—*Distributed Programming*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Additional Key Words and Phrases: Ambient calculi, mobile computation, access control systems, type systems, type safety

1. INTRODUCTION

There is a general agreement that programming languages for wide-area computing and mobile-code environments should be designed according to appropriate principles, among which distribution, location awareness, and security are the most fundamental.

Cardelli and Gordon's Mobile Ambients (MA) [Cardelli and Gordon 1998] are one of the first, and currently one of the most successful implementations of these principles into a formal calculus. Their design is centered around four basic notions: location, mobility, communication by shared location, and authorization to move based on acquisition of

Authors' address: Michele Bugliesi and Silvia Crafa: Università Ca' Foscari di Venezia, Dipartimento di Informatica, Via Torino 155, 30172, Mestre (VE), Italy. E-mail:{michele,silvia}@dsi.unive.it, Web: http://www.dsi.unive.it/~{michele,silvia}. Giuseppe Castagna: LIENS École Normale Supérieure 45, rue d'Ulm 75005 Paris France. E-mail:Giuseppe.Castagna@ens.fr, Web: http://www.di.ens.fr/ ∼castagna.

names and capabilities. The ability or inability to cross boundaries, which is conferred by the capabilities in and out, is at the core of the security model underlying MA. Permission to cross ambient boundaries is given by making the name available to the clients requesting access. Names are thus viewed as passwords, or cryptokeys: when embedded in a capability, an ambient name provides the pass that enables the access to, or else the cryptokey that discloses the contents of that ambient.

While MA's model of security is suggestive, and powerful for its simplicity, it does not appear to be fully adequate for modeling realistic access control policies. Security in MA entirely depends on the ability by the naming-based authorization mechanism to filter out unwanted clients: an authorization breach could grant malicious agents full access to all the resources located inside the ambient boundary.

An assessment of security and access control in ambient-based calculi is the main motivation for the present paper. The focus of our analysis is on *mandatory* (i.e., system-wide) *access control* policies (MAC) within a multilevel security system. In particular, the emphasis is on the specific aspects of MAC policies related to confidentiality, and their different implementations as *military* security (no read-up, no write-down) and *commercial* security (no read-up, no write-up). For other calculi of mobility in the literature, notably for Dπ [Riely and Hennessy 1998] and KLAIM [De Nicola et al. 1998], an in-depth study of these aspects has already been conducted [Hennessy and Riely 2002b; 2002a; De Nicola et al. 2000; De Nicola et al. 2000]. Instead, to our knowledge, no attempt in this direction has been made for MA-based calculi.

Our analysis, detailed in the first part of the paper, points out the shortcomings of MA as a formal basis for reasoning about these concepts. In fact, the main difficulties come far ahead of any formal reasoning, because the very meaning of basic notions such as "read access" and "write access" by subjects to objects is difficult to grasp and characterize when looked at from within MA.

To overcome these difficulties, we introduce a variant of Mobile Ambients, named *Boxed Ambients* (BA). Boxed Ambients inherit from MA the primitives in and out for mobility, but not open, and introduce direct primitives for communication across ambient boundaries, between parent and child. This new form of communication fits the design principles of MA, and complements the existing constructs for ambient mobility, and local exchanges, with finer-grained, and more effective, mechanisms for ambient interaction. The resulting calculus retains the computational flavor of MA and the elegance of its formal presentation. On the other hand, the new communication model preserves the flexibility of typed communications from MA, while providing more effective means for reasoning about access control policies.

We study two versions of the calculus, based on synchronous and asynchronous communication, respectively. Interestingly, the new model of communication sheds new insight into the relationship between the two forms of interaction. In particular, we show that classical encodings of the asynchronous model in terms of the synchronous one do not carry over to calculi that combine non-local exchanges and dynamic system reconfiguration based on mobility. We complement the definition of the calculus with a study of different type systems. A first type system provides standard safety guarantees for communication. A second type system enhances the typing of mobility and develops a new typing technique, based on different typing "modes" for processes, in which processes and their continuations may have different types while still preserving subject reduction. A last type system combines the new technique with a richer class of types to provide for the static detection of violations of MAC policies in a multilevel security environments. All the type systems, in particular the access control type system are designed, and proved sound, for both the synchronous and the asynchronous versions of the calculus. Remarkably, the moded typing system is initially motivated by the synchronous semantics but then proves equally effective for the asynchronous calculus that we eventually adopt in our discussion of access control.

Plan. Section 2 presents our analysis of security and access control in MA. Section 3 introduces the calculus of Boxed Ambients. Section 4 details encodings of additional primitives for communications on named channels (BA relies on anonymous channels). Section 5 introduces the basic type system for the calculus. Section 6 compares the typing systems of BA and MA with respect to mobility and communication. Section 7 develops an enhanced type system based on the technique named "moded typing". Section 8 studies the asynchronous version of the calculus. Section 9 develops a sound typing system for static access control, and illustrates its use with several BA programs. Section 10 studies a more extensive example: in particular, it shows that the access control typing system can effectively be employed to specify (and statically enforce) diverse and powerful security policies for a simple, but non-trivial, distributed language. Section 11 compares our approach with related work, and Section 12 concludes with final remarks. Two separate appendices collect the typing rules and the proofs of subject reduction and type soundness.

The paper integrates and extends the results reported in [Bugliesi et al. 2001a] and [Bugliesi et al. 2001b].

2. MOTIVATIONS FOR BEING *BOXED*

Mobile Ambients are named process of the form $a[P]$ where a is a name and P a process. Processes can be composed in parallel, as in $P \mid Q$, be replicated as in $!P$, exercise a capability, as in $M.P$, declare local names as in $(\nu a)P$, or simply do nothing as in $\mathbf{0}$. Ambients may be nested to form a tree structure that can be dynamically reconfigured by exercising the capabilities in, out and open. In addition, ambients and processes may communicate. Communication is anonymous, and happens inside ambients. The configuration $(x)P \mid \langle M \rangle$ represents the parallel composition of two processes, the output process $\langle M \rangle$ that "drops" the message M, and the input process $(x)P$ that reads the message M and continues as $P\{x := M\}$, that is P where every free occurrence of x has been substituted with M. The open capability has a fundamental interplay with communication: in fact, communication results from a combination of mobility and opening control. To exemplify, the synchronization between the input process $(x)P$ and the output $\langle M \rangle$ in the system $(x)P \mid \mathsf{open}\ b \mid b[\langle M \rangle \mid Q]$ is enabled by exercising the capability $\mathsf{open}\ b$ to unleash the message $\langle M \rangle$.

While fundamental in MA to enable communication across ambient boundaries, the open capability appears to bring about serious security concerns in distributed applications.

Consider a scenario in which a process P running on host h downloads an application program Q from some other host over the network. This situation can be represented by the configuration $a[\mathsf{in}\ h.Q] \mid h[P]$, where Q is included in the pilot ambient a which is routed to h in response to the download request from P. As a result of a exercising the capability $\mathsf{in}\ h$, the system evolves into the new configuration $h[a[Q] \mid P]$, where the download is completed. The application program Q may be running and computing within a, but as long as it is encapsulated into a, there is no way that P and Q can effectively interact. To enable these interactions, P will need to dissolve the transport ambient a. Dissolving a produces the new configuration $h[P \mid Q]$ where now P and Q are granted free access to each other's resources, with the obvious problem that there is no way to tell what Q may do with them. An alternative solution to the above scenario is to treat a as a *sandbox* and take the Java approach to security: P clones itself and enters the sandbox to interact with Q. Again, however, the kind of interaction between P and Q is not fully satisfactory: either they interact freely within a, or are isolated from each other.

Static or dynamic analysis of incoming code are often advocated as solutions to the above problem: incoming code must be statically checked and certified prior to being granted access to resources and sensitive data. Various authors explore this possibility, proposing control-flow analyses [Nielson et al. 1999; Nielson and Nielson 2000; Degano et al. 2000] and type systems [Cardelli et al. 1999; Dezani-Ciancaglini and Salvo 2000; Bugliesi and Castagna 2001] for Mobile Ambients. The problem with these solutions is that they may not be always feasible in practice: the source code of incoming software may be not available for analysis, or else it may be too complex to guarantee a rigorous assessment of its behavior. By that, we do not intend to undermine or dismiss the role of static analysis: instead, we take it as a motivation to seek for new design principles and more effective uses of static analysis. One such principle for Mobile Ambients, which we advocate and investigate in this paper, is that ambient interaction should be controlled by finer-grained policies to prevent from unrestricted resource access while still providing effective communication primitives.

To motivate the point further, we discuss a simple but concrete example of access control in a multilevel security system. Multilevel security presupposes a lattice of security levels and an assignment to every subject and object of a level in this lattice. Based on these levels, a read from an object by a subject is classified as a *read-up* (respectively, *read-down*) if the level of the subject is higher than the level of the object, and similarly for a write operation. These notions cover also *indirect* accesses resulting from the composition of atomic operations: thus, for example, writing to (respectively, reading from) an ℓ-level object a piece of information read (or just coming) from an h-level (with $h > \ell$) object is considered as a write-down (respectively, read-up)[1]). Relying on this classification, one typically identifies two MAC security policies: *military* security, which forbids (both direct and indirect) read-up's and write-down's, and commercial security that forbids read-up's and write-up's.

2.1 Resource access control in multilevel security

Suppose we have a system consisting of a set of resources $\{r_1, \ldots, r_n\}$ and an agent named a that runs program P and wants to access the resources available on the system. To control the access to the resources, one would typically refer to [Department of Defense 1985] and set up a resource manager. In the Ambient Calculus the system under consideration can be represented as follows:

$$a[P] \mid m[r_1[\cdots] \mid \cdots \mid r_n[\cdots] \mid R]$$

[1] Classic security handles these cases by the so-called $\star$-property [Bell and Padula 1976; Gollmann 1999]. As a matter of fact, these references do not define precisely what a *write-down access* is; instead, they give a definition of *no-write down policy*.

Here, m is the name of the resource manager and R is the associated process. To access, say, r_i, the agent a needs to know the name m to be able to move inside the resource manager:

$$m[a[P] \mid r_1[\cdots] \mid \cdots \mid r_n[\cdots] \mid R]$$

Looking at this configuration, we notice that the process R does not have an active role in the system, as the interaction between $a[P]$ and r_i may only result from autonomous actions by either the agent or the resource (the same would be true with Levi and Sangiorgi's *Safe Ambients* [Levi and Sangiorgi 2000]: only the use of a co-action could predicate the move of a into m to the presence of the co-capability $\overline{\mathsf{in}}\ m$ in R). The role of the ambient m is therefore reduced to the role of its name: it is simply the first password required for the access. Rather, it is each of the r_i's that needs to include its own manager.

We can thus formulate the problem in simpler terms, and look directly at the case below:

Initial configuration: $a[P] \mid r[R \mid \langle M\rangle]$

R is the manager for r, and M is the content: for the purpose of the example we assume that the content is a value the agent wants to read.

Having defined the problem, we now look at different ways to attack it in MA and discuss their implications in terms of the security models introduced above.

2.1.1 *First solution: agent dissolution.* A first solution is based on the following protocol proposed by [Cardelli and Gordon 1998]. In order to access r, a first enters r:

Enter: $r[R \mid \langle M\rangle \mid a[P]]$

Now, the idea of the protocol is that the manager R should be the process $!\mathsf{open}\ p$, which unleashes authorized clients that entered the resource within a pilot ambient named p. In other words, the protocol requires the client to know the name of the resource, as well the name of the "port" p used for the access. Thus, the agent would first rename itself to p to comply with the rules of the protocol, and then enter: if the access to r is a read, the agent will contain a reading process. Thus, after renaming, the new configuration would be as follows:

Renaming: $r[!\mathsf{open}\ p \mid \langle M\rangle \mid p[(x)P]]$

Finally, the resource manager enables the read, by opening p:

Read Access: $r[!\mathsf{open}\ p \mid \langle M\rangle \mid p[(x)P]] \longrightarrow r[!\mathsf{open}\ p \mid \langle M\rangle \mid (x)P]$

The protocol is elegant and robust: there are two passwords the agent needs to know, the resource name r and the name of the port p. There are, however, a number of unsatisfactory aspects to it.

A first reason for being unsatisfied with the protocol is that it is hardly realistic to assume that agents willing to read a value should be prepared to be dissolved. A second problem is that opening $p[P]$ may be upsetting to the resource manager, or else to the resource itself, because there is no telling what P might do once unleashed. For what we know, the contents of p could very well be the process $N.P$, with N a path of in or out capabilities. Unleashing this process inside r could thus result in r being carried away to possibly hostile locations, or otherwise being made unavailable to forthcoming clients.

Further problems arise when we try to classify the protocol according to the MAC security principles. As we noted, the action in the protocol that eventually enables the read is taken by the resource manager, which opens the incoming agent. In other words, it is the last step of the protocol that effectively determines the access to the resource, and since the process enclosed in p is an input process, it is classified as a read access (had p contained an output, this would have been a write access). In multilevel security, it would then be possible to further classify the access according to the security levels associated with r and p, and use that definition to enforce either the military or the commercial security policy.

However, while this form of classification fits the protocol, it becomes rather artificial when applied to the primitives of the calculus. Indeed, saying that $\mathsf{open}\ p \mid p[\,P\,]$ is a read (or write) by P is rather counter-intuitive, as $p[\,P\,]$ undergoes the action rather than actively participating into it. The problem is that the protocol is entirely dependent on the effects of open, but when exercised to enable a read/write request, open exchanges the roles of the two participants in the request, as it is the subject, rather than the object, that is accessed (in fact, opened). As a result, the notion of read/write access becomes rather artificial.

2.1.2 *Second solution: resource dissolution.*. An alternative solution can be obtained by a change of perspective. One could devise a different protocol where the active role of the subject is rendered by a combination of open and input/output. Thus, for instance, the process $\mathsf{open}\ r.(x)P$ could be interpreted, in the protocol, as a read request on r. This might work reasonably for read requests, even though the interpretation is still weak, as the access has also the side-effect of dissolving the resource. Even weaker would be the interpretation of $\mathsf{open}\ r.\langle M\rangle$ as a write: after dissolving r the output $\langle M\rangle$ really has nothing to do with a write on r.

2.1.3 *Third solution: agents and messengers.*. To avoid indiscriminate dissolution upon read and write, Cardelli and Gordon [Cardelli and Gordon 1998] suggest a different approach, based on a protocol in which agents use special ambients acting as messengers to communicate. The idea is to envisage two classes of messengers:

- *output messenger*: $o[\,N.\langle M\rangle\,]$, where N is a path to the location where the message M can be delivered
- *input messengers*: $i[\,N.(x)o[\,N^{-1}.\langle x\rangle\,]\,]$, where N is the path to the location where a value can be read. Once read, the messenger goes back to its original location (we informally use N^{-1} to denote the inverse path of M) where it delivers the value just read.

Thus, a read on r would be encoded by a protocol based on the following initial configuration:

$$a[\,\mathsf{open}\ o.(x)P \mid i[\,\mathsf{out}\ a.\mathsf{in}\ r.(x)o[\,\mathsf{out}\ r.\mathsf{in}\ a.\langle x\rangle\,]\,]\,] \;\mid\; r[\,!\mathsf{open}\ i \mid \langle M\rangle\,]$$

The protocol still requires cooperation by the resource manager, which is expected to open the input messenger. Also, looking at the primitive reductions, it would still be counter-intuitive to say that $\mathsf{open}\ i \mid i[\,P\,]$ is a read access: the classification would be more realistic, were it possible to identify i as input-messenger within r. Unfortunately, there is no syntactic way to tell messengers from ambients playing the role of "pure" agents, nor is there any syntactic way to detect "illegal" attempts to dissolve "pure" agents. Defining a notion of access, and attempting a syntactic classification would therefore still be problematic, if at all possible.

Types could be appealed to for more satisfactory solution. One could devise a typed partition of ambients into agents (i.e. ambients that cannot be dissolved) and messengers (as above). Based on the typed classification and on an assignment of security levels, it would then be possible to classify access requests according to MAC policies. There would be only one remaining problem, which can be observed by examining the protocol structure and evolution. From the initial configuration:

$$a[\,\mathsf{open}\ o.(x)P \mid i[N.(x)o[N^{-1}.\langle x\rangle\,]\,]\,] \mid r[\,!\mathsf{open}\ i \mid \langle M\rangle\,]$$

via a sequence of reductions the input messenger reaches its destination, it is opened there, and consumes M. At this stage, the structure of the system is:

$$a[\,\mathsf{open}\ o.(x)P\,] \mid r[\,!\mathsf{open}\ i \mid o[N^{-1}.\langle M\rangle\,]\,]$$

This is the encoding of a write by r to a. In other words, a read by a includes a write by r: if the former is, say, a read-up, then the latter is a write-down. In other words, the protocol has somehow the effect of merging read-up's and write-down's, and dually, write-up's and read-down's. Therefore, military security could still be accounted for with this approach, while commercial security could not.

2.2 Summary and Assessment

The survey of solutions we have given might still be incomplete, even though we do not see any fundamentally different approach to attack the problem. As to the approaches we have presented, none of them is entirely satisfactory. Some of them appear artificial, since essential intuition is lost in the encoding of the protocol (§ 2.1.1,§ 2.1.2). In others, the intuition is partially recovered but only at the expenses of failing to provide full account for both military and commercial security (§ 2.1.3).

Summarizing, we may certainly say that the Ambient Calculus *enables* access control, in that it provides constructs for encoding access protocols. Depending on the protocol, types may help define and check the desired security policies. On the other hand, the calculus *does not in itself support* these mechanisms and policies, as it does not provide built-in facilities to make it convenient or natural to reason about them. As we showed, the reasoning is possible at the level of access *protocols*, but when we look at the access *primitives* there appears to be no general principle to which one can steadily appeal. We are thus in need for different, finer-grained, constructs for ambient interaction and communication. The new constructs should be designed carefully, so as to complement the existing restrictions on ambient mobility based on authorization, without breaking them. In other words, the access to remote resources should still require mobility, hence authorization: local access, instead, could be made primitive.

To see how that can be accomplished, let us consider once more the protocol of § 2.1.3, based on messengers. We can re-state it equivalently as follows:

$$a[\,\mathsf{in}\ r.i[\,\mathsf{out}\ a.(x)o[\,\mathsf{in}\ a.\langle x\rangle\,]\,] \mid \mathsf{open}\ o.(x)\mathsf{out}\ r.P\,] \;\mid\; r[\,!\mathsf{open}\ i \mid \langle M\rangle\,]$$

In other words, it is now the agent that is responsible for the moves needed to reach the resource, while the messenger just makes the in and out moves needed for the access. After the move of a into r, and of i out of a, the structure of the system (disregarding a and replication) is the following: $r[\,\mathsf{open}\ i \mid \langle M\rangle \mid i[\,(x)Q\,]\,]$. This is where the read takes place. Now, instead of coding it, via open, we can make it primitive, and do without open. If we denote with $(x)^{\uparrow}$ input from the enclosing ambient, the read access is simply: $r[\,\langle M\rangle \mid i[\,(x)^{\uparrow}Q\,]\,]$. But then, the whole protocol can be simplified: $a[\,\mathsf{in}\ r.(x)^{\uparrow}P\,] \mid r[\,\langle M\rangle\,]$. The choice of the communication primitives of Boxed Ambients, described next, are based on these observations.

3. BOXED AMBIENTS

Boxed Ambients are essentially Mobile Ambients that cannot be opened. Processes in the new calculus communicate, as in the Ambient Calculus, on anonymous channels inside ambients. In addition, to compensate for the absence of open, processes are equipped with primitives for communication across ambient boundaries, between parent and children. Syntactically, this is achieved by means of tags specifying the *location* , i.e., the ambient, where the communication takes place.

3.1 Syntax

Table I Boxed Ambients

Expressions

M	$::=$	$a-q$	names
	$\mid$	$x-z$	variables
	$\mid$	$\mathsf{in}\ M$	enter M
	$\mid$	$\mathsf{out}\ M$	exit M
	$\mid$	$M.M$	path

Locations

η	$::=$	M	child
	$\mid$	$\uparrow$	parent
	$\mid$	$\star$	local

Processes

P	$::=$	$\mathbf{0}$	stop
	$\mid$	$M.P$	action
	$\mid$	$(\nu n)P$	restriction
	$\mid$	$P \mid P$	composition
	$\mid$	$M[P]$	ambient
	$\mid$	$!P$	replication
	$\mid$	$(x_1,\ldots,x_k)^{\eta}P$	input, $k \geqslant 0$
	$\mid$	$\langle M_1,\ldots,M_k\rangle^{\eta}P$	output, $k \geqslant 0$

The untyped syntax of the polyadic calculus is summarized in Table I. It includes two syntactic categories, *expressions* and *processes*. Expressions, ranged over by M,N, include *names*, *variables* and *capabilities*. We presuppose two mutually disjoint sets for variables and names. Variables are ranged over by letters toward the end of the alphabet, typically x,y,z, while the remaining letters $a-q$ are reserved for names. The capabilities in and out enable movement and can be composed into non-empty *paths*.

Processes, ranged over by P,Q,R,S, are built from the constructors of *inactivity*, *parallel composition*, *replication* and *restriction* inherited from the π-calculus, and from four additional operators: *prefix* $M.P$, anonymous (polyadic) *input* $(\tilde{x})^{\eta}P$ and *output* $\langle\tilde{M}\rangle^{\eta}P$ and *ambient* $M[P]$. The notation $\tilde{M}$ indicates a tuple of messages $M_1,\ldots,M_k$, and similarly $\tilde{x}$ is short for $x_1,\ldots,x_k$. When $k=0$ the input/output prefixes allow synchronization without exchange of values. The superscript $\star$ denoting local communication, is omitted. Similarly, we often omit trailing and isolated occurrences of $\mathbf{0}$, writing M instead of $M.\mathbf{0}$, and $n[\ ]$ instead of $n[\mathbf{0}]$. The input operator $(\tilde{x})^{\eta}P$ is a binder for the *variables* $\tilde{x}$, whereas the restriction operator $(\nu n)P$ binds the *name* n: in both cases the scope of the binder is P. As it is customary, terms that are α-convertible are considered identical. The notions of *free names* and *free variables* of a process, noted $fn(P)$ and $fv(P)$ respectively, arise as expected (see Table II), and so does the definition of *capture free* substitution $P\{\tilde{x} := \tilde{M}\}$, that is defined only if $\tilde{x}$ and $\tilde{M}$ are of the same arity. A process is *closed* if it contains no free variables.

Table II Free names and free variables

Free Names	Free Variables
$fn(\star) = fn(\uparrow) = \emptyset$	$fv(\star) = fv(\uparrow) = \emptyset$
$fn(n) = \{n\}$	$fv(n) = \emptyset$
$fn(x) = \emptyset$	$fv(x) = \{x\}$
$fn(\mathsf{in}\ M) = fn(\mathsf{out}\ M) = fn(M)$	$fv(\mathsf{in}\ M) = fv(\mathsf{out}\ M) = fv(M)$
$fn(M.M') = fn(M) \cup fn(M')$	$fv(M.M') = fv(M) \cup fv(M')$
$fn(\mathbf{0}) = \emptyset$	$fv(\mathbf{0}) = \emptyset$
$fn(M.P) = fn(M[\,P\,]) = fn(M) \cup fn(P)$	$fv(M.P) = fv(M[\,P\,]) = fv(M) \cup fv(P)$
$fn((\nu n)P) = fn(P) \setminus \{n\}$	$fv((\nu n)P) = fv(P)$
$fn(P_1 \mid P_2) = fn(P_1) \cup fn(P_2)$	$fv(P_1 \mid P_2) = fv(P_1) \cup fv(P_2)$
$fn(!P) = fn(P)$	$fv(!P) = fv(P)$
$fn((\tilde{x})^{\eta}P) = fn(P) \cup fn(\eta)$	$fv((\tilde{x})P) = fv(P) \cup fv(\eta) \setminus \{\tilde{x}\}$
$fn(\langle \tilde{M} \rangle^{\eta}P) = fn(P) \cup fn(\tilde{M}) \cup fn(\eta)$	$fv(\langle \tilde{M} \rangle^{\eta}P) = fv(P) \cup fv(\tilde{M}) \cup fv(\eta)$

3.2 Operational Semantics

The operational semantics of the calculus is defined, as customary, in terms of structural congruence and reduction relation. Both these relations are summarized in Table III. Structural congruence is the least congruence relation that satisfies the (Struct) laws in Table III. The first group of laws are the familiar monoidal laws for $\mid$ and $\mathbf{0}$. The second group of laws is inherited from the Ambient Calculus.

The remaining laws in Table III define the reduction relation, which applies to closed processes. Ambient mobility is governed by the rules (enter) and (exit), inherited from Mobile Ambients. The rule for (local), and the structural rules (struct) and (context) also are defined as in MA. The remaining four rules define the reduction for parent-child exchange.

The choice of the reductions for parent-child exchanges, and the resulting model of communication is inspired to Castagna and Vitek's *Seal Calculus* [Vitek and Castagna 1999], from which Boxed Ambients also inherit the two principles of *locality* and *mediation*. Locality means that communication resources are *local* to ambients, and message exchanges result from explicit read and write requests on those resources. In particular, the input prefix $(x)^n$ can be seen as a request to read from the anonymous channel located into the child n. In fact, given the anonymous nature of channels, $(x)^n$ can equivalently be seen as an access to the ambient n. Dually, $\langle M \rangle^{\uparrow}$ can be interpreted as write request to the parent ambient (equivalently, its local channel). Mediation implies that remote communication, e.g. between sibling ambients, is not possible: it either requires mobility, or intervention by the ambients' parent. The implementation of these two principles based on the the term-level constructs we have introduced for parent-child communication has a number of interesting consequences, that we discuss next.

3.2.1 *Communication and access control.*. Parent-child communication yields flexible support for programming access control policies. If we take the access control problem of Section 2.1 we now have a fairly elegant solution, in which we also recover a role for the

Table III Structural Equivalence and Reduction

Structural Congruence

(Struct Monoid)	$P \mid Q \equiv Q \mid P, \quad P \mid (Q \mid R) \equiv (P \mid Q) \mid R, \quad P \mid \mathbf{0} \equiv P$			
(Struct Res Dead)	$(\nu n)\mathbf{0}$	$\equiv$	$\mathbf{0}$	
(Struct Res Res)	$(\nu n)(\nu m)P$	$\equiv$	$(\nu m)(\nu n)P$	$m \neq n$
(Struct Path Assoc)	$(M.M').P$	$\equiv$	$M.(M'.P)$	
(Struct Res Par)	$(\nu n)(P \mid Q)$	$\equiv$	$P \mid (\nu n)Q$	$n \notin fn(P)$
(Struct Repl)	$!P$	$\equiv$	$!P \mid P$	
(Struct Res Amb)	$(\nu n)m[P]$	$\equiv$	$m[(\nu n)P]$	$n \neq m$

Mobility

(enter)	$a[\mathsf{in}\ b.P \mid Q] \mid b[R]$	$\longrightarrow$	$b[a[P \mid Q] \mid R]$
(exit)	$a[b[\mathsf{out}\ a.P \mid Q] \mid R]$	$\longrightarrow$	$b[P \mid Q] \mid a[R]$

Communication

(local)	$(\tilde{x})P \mid \langle\tilde{M}\rangle Q$	$\longrightarrow$	$P\{\tilde{x} := \tilde{M}\} \mid Q$
(input n)	$(\tilde{x})^n P \mid n[\langle\tilde{M}\rangle Q \mid R]$	$\longrightarrow$	$P\{\tilde{x} := \tilde{M}\} \mid n[Q \mid R]$
(input $\uparrow$)	$n[(\tilde{x})^{\uparrow} P \mid Q] \mid \langle\tilde{M}\rangle R$	$\longrightarrow$	$n[P\{\tilde{x} := \tilde{M}\} \mid Q] \mid R$
(output n)	$n[(\tilde{x})P \mid Q] \mid \langle\tilde{M}\rangle^n R$	$\longrightarrow$	$n[P\{\tilde{x} := \tilde{M}\} \mid Q] \mid R$
(output $\uparrow$)	$(\tilde{x})P \mid n[\langle\tilde{M}\rangle^{\uparrow} Q \mid R]$	$\longrightarrow$	$P\{\tilde{x} := \tilde{M}\} \mid n[Q \mid R]$

Structural Rules

(struct)	$P \equiv Q, \longrightarrow\ R,\ R \equiv S \ \Rightarrow\ P \longrightarrow S$
(context)	$P \longrightarrow Q \ \Rightarrow\ \mathbf{E}\{P\} \longrightarrow \mathbf{E}\{Q\}$

Evaluation Context $\quad \mathbf{E} ::= - \mid (\nu n)\mathbf{E} \mid P \mid \mathbf{E} \mid n[\mathbf{E}]$

resource manager m. Consider again the configuration

$$m[a[P] \mid R \mid r_1[\cdots] \mid \cdots \mid r_n[\cdots]]$$

where now all ambients are boxed and a has entered the resource manager. The process R may act as a mediator between a and the resources. For instance, R could be defined as the parallel composition $R_1 \mid \cdots \mid R_n$ where each R_i is the process $!(x)\langle x\rangle^{r_i}$, for $i \in 1..n$, each waiting for upward output from a and forwarding it to the ith resource. Some of the R_i's could be less generous with the agent, and ignore upward input from a to request read access on a instead: $!(x)^a\langle x\rangle^{r_i}$. Should any of the r_i's be made non-accessible, one would simply define $R_i = \mathbf{0}$. Of course, different definitions of R are possible. For instance, one could define R as the process $!(x,r)\langle x\rangle^r$, that waits for upward requests from a to write on one of the r_i's, and forwards this request to the corresponding resource.

3.2.2 *Communication and typing.*. The model of communication also eases the design of type systems providing precise accounts of ambient behavior. As we show in Section 5, a rather simple structure of types suffices for that purpose. Ambient and process types are defined as two-place constructors describing the types of the exchanges that may take place locally, and with the enclosing context. Interestingly, this simple type structure is all that is needed to gain full control of ambient interaction. This is a consequence of (i) there being no way for ambients to communicate directly across more than one boundary, and (ii) communication being the only means for ambient to interact. To exemplify, consider the following configuration:

$$(x)^p P \mid p[\,\langle M\rangle \mid (x)Q \mid q[\,\langle N\rangle^{\uparrow}\,]\,]$$

The top-level makes a downward request to read p's local value M, while ambient q makes an upward request to write the value N to its parent. The downward input $(x)^p P$ may only synchronize with the output $\langle M\rangle$ local to p. Instead, $(x)Q$ may nondeterministically synchronize with either output. Of course, type safety requires that M and N be of the same type. Interestingly, however, exchanges of different types may take place within the same ambient (or at top-level) without type confusion, as within the following ambient n:

$$n[\,(x)^p P \mid (x)^q Q \mid p[\,\langle M\rangle\,] \mid q[\,\langle N\rangle\,]\,]$$

The two values M and N are local to p and q, and may very well have different types: there is no risk of type confusion, as $(x)^p P$ reads from p, while $(x)^q Q$ reads from q. Types may also be employed to complement the term-level support for access control. By embedding security levels in types, a type system may be defined to enforce MAC security policies in rather natural way (cf. Section 9).

3.2.2.1 *Communication, distribution and location awareness.*. The new constructs for communication fit well the principles of distribution and location awareness distinctive of Mobile Ambients, according to which remote communication should require mobility (and ambient opening), and mobility, in turn, should require authorization (i.e. possession of capabilities). Our semantics adds to this a new possibility of exchanging values, across ambient boundaries. Remarkably, however, the new form of communication takes place across just one one boundary, separating parent and child, while communication between siblings still requires mobility, as in MA.

From a design perspective, our model of communication refines the notion of locality from MA into that of *proximity*, and allows synchronization between processes that are *contiguous*, i.e., either local or separated by one boundary. In all respects, this kind of synchronization is required for mobility as well, and needs to be assumed either implicitly, as in MA, or explicitly as in the variants of MA in which mobility is subject to the presence of co-capabilities [Levi and Sangiorgi 2000; Merro and Hennessy 2002; Bugliesi et al. 2002].

In addition, the constructs for communication lend themselves to be formulated in a truly asynchronous setting, in which sending output to a nested or enclosing ambient does not require synchronization. As of now, asynchronous output can either be considered as the special case of synchronous output with null continuation, or else be accounted for by introducing the following rule of structural equivalence inspired by [Boudol 1992]: $\langle M\rangle^{\eta} P \equiv \langle M\rangle^{\eta} \mid P$. These interpretations are equivalent, and both type sound, with the system of "simple" types of Section 5. Instead, the (typed) equivalence is lost with the system of "moded types" we introduce in § 7, and neither interpretation is satisfactory: the former is too restrictive, the latter is unsound. Fortunately, however, we will be able to find a formulation of reduction that reconciles asynchrony with moded typing in a sound and flexible type system. We leave a thorough discussion on this point to Section 8.

4. COMMUNICATION CHANNELS

Value exchange between BA processes can also take place on named channels: we illustrate several ways for encoding channels within the core calculus. To reason on the properties of these encodings we introduce the following definition of observational congruence for BA, that we directly inherit from MA [Cardelli and Gordon 1999a]. Given a process P, we write $P\downarrow_n$ if P has a top-level occurrence of an ambient named n, with n not restricted in P. Formally, $P\downarrow_n$ iff $P \equiv (\nu\tilde{m})(P' \mid n[P''])$ with $n \notin \tilde{m}$. Then we say that P *exhibits the name* n, written $P \Downarrow_n$, iff there exists a process Q such that $P \Longrightarrow Q$ and $Q\downarrow_n$, where $\Longrightarrow$ is the reflexive and transitive closure of $\longrightarrow$. Finally, two processes P and Q are observationally congruent, written $P \cong Q$, if $\mathbf{C}\{P\}\Downarrow_n$ iff $\mathbf{C}\{Q\}\Downarrow_n$ for all contexts $\mathbf{C}$ with $\mathbf{C}\{P\}$ and $\mathbf{C}\{Q\}$ closed.

Based on this definition, we can re-establish some useful observational equivalences. In particular, the equivalence $!P \mid\ !P \cong !P$, from the π-calculus, and the *perfect firewall* equation from MA: $(\nu n)n[P] \cong \mathbf{0}$ for $n \notin fn(P)$. Both will be useful in the remainder of this section.

4.1 π-calculus channels

We start with the asynchronous π-calculus, and then adapt our technique to handle the synchronous case. The idea is straightforward, and best illustrated with an example: two π-processes communicating over a channel c, as in $c\langle n\rangle \mid c(x)P$, may be represented in BA as follows: $c[\langle n\rangle] \mid (x)^c P$. In other words, an output on c in π is represented by an ambient named c (a buffer) holding the message which is output on the channel. An input on c, in turn, translates directly into the corresponding input prefix of BA. One problem with this simple idea is that the buffer does not go away when its value is consumed: in the example above we have $c[\langle n\rangle] \mid (x)^c P \longrightarrow c[\,] \mid P\{x := n\}$, which is unfortunate, because $c[\,] \not\cong \mathbf{0}$. The problem can be solved by having the buffer "hide itself" once the value it holds has been consumed: $(\nu p)p[\,] \mid a[\langle\tilde{M}\rangle \mathsf{in}\ p]$. Now, letting $a\{\!\!\{\tilde{M}\}\!\!\} \triangleq (\nu p)p[\,] \mid a[\langle\tilde{M}\rangle \mathsf{in}\ p]$, we define a compositional encoding of the (polyadic) asynchronous π calculus as follows:

$$
\begin{array}{lll}
\langle\!\langle (\nu a)P \rangle\!\rangle = (\nu a)\langle\!\langle P \rangle\!\rangle & \langle\!\langle !P \rangle\!\rangle = !\langle\!\langle P \rangle\!\rangle & \langle\!\langle a\langle\tilde{b}\rangle \rangle\!\rangle = a\{\!\!\{\tilde{b}\}\!\!\} \\
\langle\!\langle P \mid Q \rangle\!\rangle = \langle\!\langle P \rangle\!\rangle \mid \langle\!\langle Q \rangle\!\rangle & \langle\!\langle \mathbf{0} \rangle\!\rangle = \mathbf{0} & \langle\!\langle a(\tilde{x})P \rangle\!\rangle = (\tilde{x})^a \langle\!\langle P \rangle\!\rangle
\end{array}
$$

These definitions extend readily to the case of the synchronous π-calculus by resorting to the standard technique of representing synchronous output by means of a pair of messages (send and acknowledge). Letting $nm.P$ and $\bar{n}m.P$ denote π-calculus *synchronous* input and output on channel n, we define:

$$
\begin{array}{rcll}
\langle\!\langle \bar{a}\tilde{b}.P \rangle\!\rangle & = & (\nu r)a\{\!\!\{(\tilde{b},r)\}\!\!\} \mid (r[\,()\langle\rangle\,] \mid ()^r \langle\!\langle P \rangle\!\rangle) & r \notin fn(P) \\
\langle\!\langle a\tilde{x}.Q \rangle\!\rangle & = & (\tilde{x},y)^a \langle\rangle^y \langle\!\langle Q \rangle\!\rangle & y \notin fv(Q)
\end{array}
$$

and then extend the translation compositionally to the remaining constructs. For both these encodings one can prove that if $P \longrightarrow Q$, in the π calculus, then $\langle\!\langle P \rangle\!\rangle \longrightarrow\ \cong \langle\!\langle Q \rangle\!\rangle$ in BA.

A simpler, and more robust, translation can be obtained by extending BA with movement co-capabilities à la Safe Ambients [Levi and Sangiorgi 2000]. As for Safe Ambients, the addition of co-capabilities strengthens the algebraic theory of the calculus, and yields a richer set of congruence laws. In particular, one has $n[\,] \cong \mathbf{0}$ for all n, as there is no way that a context may test the presence of $n[\,]$ (by a move or by an exchange). One can then rely on the simple translation

$$\langle\!\langle a\langle\tilde{b}\rangle \rangle\!\rangle = a[\tilde{b}] \qquad \langle\!\langle a(\tilde{x})P \rangle\!\rangle = (\tilde{x})^a \langle\!\langle P \rangle\!\rangle$$

and extend it compositionally. The resulting translation is *operationally sound*. in the following sense: $P \longrightarrow Q$ in π implies $\langle\!\langle P \rangle\!\rangle \longrightarrow\, \cong \langle\!\langle Q \rangle\!\rangle$, and vice versa, $\langle\!\langle P \rangle\!\rangle \longrightarrow Q$, implies that $P \longrightarrow P'$ in π with $Q \cong \langle\!\langle P' \rangle\!\rangle$.

4.2 Channels as persistent resources

A different way to represent channels is to interpret them as persistent resources. This interpretation is particularly meaningful Ambient-based calculi, in which ambients may be thought of as network nodes that provide a set of fixed ports for the interaction with other nodes (cf. Section 10). This idea has a direct implementation in BA. The ambient $c[\,!(x)\langle x\rangle\,]$ represents a buffer with an unbounded number of positions: the buffer simply waits for local input and, once received, releases local output. Input and output on the buffer may then be implemented directly by the primitives for downward communication $\langle b\rangle^c$ and $(x)^c$. If we define $\texttt{channel}(c) = c[\,!(x)\langle x\rangle\,]$. then we have, as expected:

$$\texttt{channel}(c) \mid \langle b\rangle^c \mid (x)^c P \implies \;\cong\; \texttt{channel}(c) \mid P\{x := b\}$$

One may use this idea to represent persistent channels in the π calculus. To define the encoding compositionally, we associate a channel *spawner* with each input and/or output on the corresponding name.

$$\langle\!\langle a\langle\tilde{b}\rangle \rangle\!\rangle = \texttt{channel*}(a) \mid \langle\tilde{b}\rangle^a \quad \langle\!\langle a(\tilde{x})P \rangle\!\rangle = \texttt{channel*}(a) \mid (\tilde{x})^a \langle\!\langle P \rangle\!\rangle$$

where $\texttt{channel*}(c) = \,!\texttt{channel}(c)$. The presence of multiple copies of spawners and channels is harmless, as one has $\texttt{channel*}(c) \mid \texttt{channel*}(c) \cong \texttt{channel*}(c)$, while multiple copies of the channel may be garbaged collected by structural congruence:

$$\texttt{channel*}(c) \mid \texttt{channel}(c) \equiv \texttt{channel*}(c).$$

It is also worth pointing out that an implementation of channels as replicated ambients would not work in MA, because inputs and outputs could get lost in distinct copies of the channel ambient. Since our representation does not require mobility into the channel, this problem goes away in our case.

4.3 Parent-child channeled communication à la Seal Calculus

Both the techniques we have illustrated can be extended to allow value exchanges between processes located in Boxed Ambients at different nesting levels. The extension yields a set of communication protocols that are similar to those given as primitive in the Seal Calculus [Vitek and Castagna 1999]. In the Seal Calculus, one can express output prefixes of the form $c^n\langle M\rangle$ requesting a write on the channel c residing in ambient (or seal) n. Dually, the input prefix $c^{\uparrow}(x)$ denotes a read request on the channel c residing in the parent ambient. Upward output and downward input on local channels may be expressed in similar ways.

All these communication protocols can be expressed in the core calculus of Boxed Ambients: we focus on asynchronous communication, and detail the cases of downward output and upward input.

The intended reduction of a downward output is as follows:

$$c^n\langle M\rangle \mid n[\, c(x)P \mid Q\,] \;\longrightarrow\; n[\, P\{x := M\} \mid Q\,].$$

The channel c is local to n, and the outer process writes on c. There are several ways that the reduction can be captured with the existing constructs: we choose a definition that make the physical localization of c explicit. The channel c is represented as the $\texttt{channel}(c)$, the input prefix $c(x)$ as a read on c:

$$c(x)P \;\stackrel{\triangle}{=}\; \texttt{channel*}(c) \mid (x)^c P.$$

Now, however, an output on c cannot be represented directly as we did above for the π-calculus channel, because c is located into n. To capture the desired behavior we can rely on mobility:

$$c^n\langle M\rangle \;\stackrel{\triangle}{=}\; (\nu p)p[\,\mathsf{in}\ n.\mathsf{in}\ c.\langle M\rangle^{\uparrow}\,].$$

The output M is encapsulated into a "pilot" ambient p, which enters n and then c to deliver the message (the name of the pilot ambient p must be fresh). Then, the Seal Calculus process $c^n\langle M\rangle \mid n[\, c(x)P \mid Q\,]$ is encoded as follows:

$$\begin{array}{rl}
(\nu p)p[\,\mathsf{in}\ n.\mathsf{in}\ c.\langle M\rangle^{\uparrow}\,] \;\mid\; n[\,\texttt{channel*}(c) \mid (x)^c P \mid Q\,] & \\
\Longrightarrow & n[\,\texttt{channel*}(c) \mid c[\,!(x)\langle x\rangle \mid (\nu p)p[\,\mathbf{0}\,]\,] \mid P\{x := M\} \mid Q\,] \\
\cong & n[\,\texttt{channel*}(c) \mid P\{x := M\} \mid Q\,].
\end{array}$$

Remote inputs are slightly more complex, since the pilot ambient must fetch the output and bring it back. The intended reduction is $c\langle M\rangle \mid n[\,c^{\uparrow}(x)P \mid Q\,] \longrightarrow n[\,P\{x := M\} \mid Q\,]$, where the input from within n is defined as follows:

$$c^{\uparrow}(x)P \;\stackrel{\triangle}{=}\; (\nu p)p[\,\mathsf{out}\ \underline{n}.\mathsf{in}\ c.(x)^{\uparrow}\mathsf{out}\ c.\mathsf{in}\ \underline{n}.\langle x\rangle\,] \mid (x)^p P.$$

Note that the definition depends on the name n of the enclosing ambient: in a formal definition, one needs to keep track of this information, and extend the encoding of the asynchronous π calculus with the following clauses:

$$\begin{array}{lcl}
\langle\!\langle c^m\langle b\rangle \rangle\!\rangle_n & = & (\nu p)p[\,\mathsf{in}\ m.\mathsf{in}\ c.\langle b\rangle^{\uparrow}\,] \\
\langle\!\langle c^{\uparrow}\langle b\rangle \rangle\!\rangle_n & = & (\nu p)p[\,\mathsf{out}\ n.\mathsf{in}\ c.\langle b\rangle^{\uparrow}\,] \\
\langle\!\langle c^m(x)P \rangle\!\rangle_n & = & (\nu p)p[\,\mathsf{in}\ m.\mathsf{in}\ c.(x)^{\uparrow}\mathsf{out}\ c.\mathsf{out}\ m.\langle x\rangle\,] \mid (x)^p \langle\!\langle P \rangle\!\rangle_n \\
\langle\!\langle c^{\uparrow}(x)P \rangle\!\rangle_n & = & (\nu p)p[\,\mathsf{out}\ n.\mathsf{in}\ c.(x)^{\uparrow}\mathsf{out}\ c.\mathsf{in}\ n.\langle x\rangle\,] \mid (x)^p \langle\!\langle P \rangle\!\rangle_n \\
\langle\!\langle a[P] \rangle\!\rangle_n & = & a[\,\langle\!\langle P \rangle\!\rangle_a\,]
\end{array}$$

5. TYPED BOXED AMBIENTS

As we stated at the outset, one of the goals in the design of Boxed Ambients is to enable simple and effective static analyses of ambient and process behavior, while preserving the expressive power of the calculus. The definition of the type system, given in this section, proves that the design satisfies these requirements. Ambient and process types are defined simply as two-place constructors describing the types of the exchanges that may take place locally and with the enclosing context.

5.1 Types and Typed Syntax

The typed syntax is derived directly from the untyped version of the calculus by associating types with names and variables introduced by restrictions and input prefixes. Accordingly, we henceforth denote restricted processes by $(\nu n\colon A)P$ and input processes by $(\tilde{x}\colon \tilde{W})P$, where A is an ambient type and $\tilde{W}$ is a (tuple) expression type to be defined next. The relations of structural congruence and reduction extend to the typed syntax as expected.

The structure of types is defined by the following productions.

$$
\begin{array}{llcll}
\textit{Expression Types} & W & ::= & \mathsf{Amb}[E,F] & \text{ambient} \\
 & & \mid & \mathsf{Cap}[E] & \text{capability} \\[1em]
\textit{Exchange Types} & E,F & ::= & \mathsf{shh} & \text{no exchange} \\
 & & \mid & W_1 \times \cdots \times W_k & \text{tuple, } k \geqslant 0 \\[1em]
\textit{Process Types} & T & ::= & \mathsf{Pro}[E,F] & \text{composite exchange}
\end{array}
$$

The structure of types is superficially similar to that of companion type systems for the Ambient Calculus [Cardelli and Gordon 1999b; Cardelli et al. 1999]. In [Cardelli and Gordon 1999b] (and the core system of [Cardelli et al. 1999]), ambients and processes have types of the form $\mathsf{Amb}[E]$ and $[E]$, respectively, where E denotes the type of local exchanges, and the typing rules ensure that processes with type $[E]$ may only be enclosed in ambients of type $\mathsf{Amb}[E]$. Capabilities, in turn, have types of the form $\mathsf{Cap}[E]$, and the type system guarantees that exercising a capability with this type will only unleash (i.e. open ambient enclosing) processes with E-exchanges. Instead, our types are interpreted as follows:

- $\mathsf{Amb}[E,F]$: ambients that enclose processes of type $\mathsf{Pro}[E,F]$,
- $\mathsf{Cap}[E]$: capabilities exercised within ambients with E upward exchanges,
- $\mathsf{Pro}[E,F]$: processes with local and upward exchanges of types E and F, respectively.

Notice that capability types disregard the local exchanges of the ambients where they are exercised: this is possible because exercising a capability within an ambient may only cause that ambient to move, and the safety of ambient mobility may be established regardless of the ambient's local exchanges. As for process types, we give the intuitions about composite exchange with few examples:

- $(x\colon W)\langle x\rangle : \mathsf{Pro}[W,\mathsf{shh}]$. W is exchanged (read and written) locally, and there is no upward exchange.
- $(x\colon W)^{\uparrow}\langle x\rangle^{n} : \mathsf{Pro}[\mathsf{shh},W]$. W is exchanged (i.e. read from) upward, and then written to ambient n. There is no local exchange, hence the type shh as the first component of the process type. For the typing to be derivable, one needs $n : \mathsf{Amb}[W,E]$ for some exchange type E.
- $(x\colon W)^{\uparrow}(y\colon W')(\langle x\rangle^{n} \mid \langle y\rangle) : \mathsf{Pro}[W',W]$. W is exchanged (read from) upward, and then forwarded to ambient n, while W' is exchanged (read and written) locally. Again, for the typing to be derivable, one needs $n : \mathsf{Amb}[W,E]$ for some exchange type E.
- $(x\colon W)\langle x\rangle^{\uparrow} : \mathsf{Pro}[W,W]$. W is read locally, and written upward.

These simple examples give a flavor of the flexibility of the communication primitives: like mobile ambients, boxed ambients are "places of conversation", but unlike ambients they allow more than just one "topic" of conversation. Specifically, every ambient may exchange values of different types with any of its children, as long as the exchange is directed from the ambient to the children. Instead, upward communication is subject to more constraints: all the children must agree on the (unique) type of exchange they may direct to their parent.

5.2 Typing Rules

The judgments of the type system have two forms: $\Gamma \vdash M : W$, read "*expression M has type W under* Γ", and $\Gamma \vdash P : T$, read "*process P has type T under* Γ", where Γ is a type environment mapping names and variables into types. In addition, we introduce the following definition of subtyping.

DEFINITION SUBTYPING. *Exchange subtyping, noted $\leqslant$, is the smallest preorder relation over exchange types such that* $\mathsf{shh} \leqslant E$ *for every exchange type E. Process subtyping is the smallest preorder relation $\leqslant$ over process types such that* $\mathsf{Pro}[E,F] \leqslant \mathsf{Pro}[E',F]$ *if and only if* $E \leqslant E'$.

The intuition for subtyping is simple: a silent exchange can always be subsumed by a non-silent exchange. However, to ensure type soundness, the subtyping relations must be defined and used with care in the typing rules. Remarkably, the definition disallows seemingly harmless forms of *in depth* subtyping over capability types, such as $\mathsf{Cap}[\mathsf{shh}] \leq \mathsf{Cap}[E]$, and further relations over process types, like $\mathsf{Pro}[E,\mathsf{shh}] \leqslant \mathsf{Pro}[E,F]$. In addition, the typing rules will allow uses of subsumption only in conjunction with process subtyping, *not* with exchange subtyping. To motivate these restrictions, we first need to introduce the typing rules. Below, we discuss the most interesting ones.

5.2.1 *Typing of Expressions.* Rules (IN) and (OUT), below, define the constraints for ambient mobility to be safe, and explain why capability types are built around a single component.

$$\text{(IN)}\ \frac{\Gamma \vdash M : \mathsf{Amb}[F,E] \quad F' \leqslant F}{\Gamma \vdash \mathsf{in}\ M : \mathsf{Cap}[F']} \qquad \text{(OUT)}\ \frac{\Gamma \vdash M : \mathsf{Amb}[E,F] \quad F' \leqslant F}{\Gamma \vdash \mathsf{out}\ M : \mathsf{Cap}[F']}$$

The intuition is as follows: take a capability, say $\mathsf{in}\ n$, and suppose that this capability is exercised within ambient, say, m. If m has upward exchanges of type F', then $\mathsf{in}\ n : \mathsf{Cap}[F']$. Now, if $n : \mathsf{Amb}[F,E]$, in order for the move of m into n to be safe, one must ensure that the type F of the local exchanges of n be equal to the type F' of the upward exchanges of m. In fact, the typing can be slightly more flexible, for if m has no upward exchange, then $F' = \mathsf{shh} \leqslant F$, and m may safely move into n. Dual reasoning applies to the (OUT) rule: the upward exchanges of the exiting ambient must have type $\leqslant$-compatible with the type of the upward exchanges of the ambient being exited. The (PATH) rule has the same format as the corresponding rule in type systems for Mobile Ambients, namely:

$$\text{(PATH)}\ \frac{\Gamma \vdash M_1 : \mathsf{Cap}[F] \quad \Gamma \vdash M_2 : \mathsf{Cap}[F]}{\Gamma \vdash M_1.M_2 : \mathsf{Cap}[F]}$$

5.2.2 *Typing of Processes*

(DEAD)
$$\frac{}{\Gamma \vdash \mathbf{0} : [E,F]}$$

(NEW)
$$\frac{\Gamma, n : W \vdash P : [E,F]}{\Gamma \vdash (\nu n{:}W)P : [E,F]}$$

(PARALLEL)
$$\frac{\Gamma \vdash P : \mathsf{Pro}[E,F] \quad \Gamma \vdash Q : \mathsf{Pro}[E,F]}{\Gamma \vdash P \mid Q : \mathsf{Pro}[E,F]}$$

(SUBSUM PROC)
$$\frac{\Gamma \vdash P : T \quad T \leqslant T'}{\Gamma \vdash P : T'}$$

(REPLICATION)
$$\frac{\Gamma \vdash P : \mathsf{Pro}[E,F]}{\Gamma \vdash\, !P : \mathsf{Pro}[E,F]}$$

(PREFIX)
$$\frac{\Gamma \vdash M : \mathsf{Cap}[F] \quad \Gamma \vdash P : \mathsf{Pro}[E,F]}{\Gamma \vdash M.P : \mathsf{Pro}[E,F]}$$

(AMB)
$$\frac{\Gamma \vdash M : \mathsf{Amb}[E,F] \quad \Gamma \vdash P : \mathsf{Pro}[E,F]}{\Gamma \vdash M[P] : \mathsf{Pro}[F,G]}$$

(DEAD), (NEW), (PARALLEL), (REPLICATION) and the subsumption rule are standard. In the (PREFIX) rule, the typing of the capability M ensures, via the (IN), (OUT), and (PATH) rules introduced earlier, that each of the ambients being traversed as a result of exercising M have local exchanges of type compatible with the upward exchanges of the current ambient (that is, the one moved by M). The rule (AMB) establishes the conditions that must be satisfied for P to be safely enclosed in M: specifically, the exchanges of P must have the same types E and F as the exchanges declared for M. In fact, P could be locally silent, and the typing of $M[P]$ be derivable from $\Gamma \vdash P : \mathsf{Pro}[\mathsf{shh},F]$ by subsumption. In addition, if $\Gamma \vdash M : \mathsf{Amb}[E,\mathsf{shh}]$, and $\Gamma \vdash P : \mathsf{Pro}[E,\mathsf{shh}]$, then by (AMB) $\Gamma \vdash M[P] : \mathsf{Pro}[\mathsf{shh},G]$, and then by subsumption $\Gamma \vdash M[P] : \mathsf{Pro}[F,G]$, for any F and G.

(INPUT $\star$)
$$\frac{\Gamma, \tilde{x}{:}\tilde{W} \vdash P : \mathsf{Pro}[\tilde{W},E]}{\Gamma \vdash (\tilde{x}{:}\tilde{W})P : \mathsf{Pro}[\tilde{W},E]}$$

(OUTPUT $\star$)
$$\frac{\Gamma \vdash \tilde{M} : \tilde{W} \quad \Gamma \vdash P : \mathsf{Pro}[\tilde{W},E]}{\Gamma \vdash \langle \tilde{M} \rangle P : \mathsf{Pro}[\tilde{W},E]}$$

(INPUT M)
$$\frac{\Gamma \vdash M : \mathsf{Amb}[\tilde{W},E] \quad \Gamma, \tilde{x}{:}\tilde{W} \vdash P : T}{\Gamma \vdash (\tilde{x}{:}\tilde{W})^M P : T}$$

(OUTPUT M)
$$\frac{\Gamma \vdash N : \mathsf{Amb}[\tilde{W},E] \quad \Gamma \vdash \tilde{M} : \tilde{W} \quad \Gamma \vdash P : T}{\Gamma \vdash \langle \tilde{M} \rangle^N P : T}$$

(INPUT $\uparrow$)
$$\frac{\Gamma, \tilde{x}{:}\tilde{W} \vdash P : \mathsf{Pro}[E,\tilde{W}]}{\Gamma \vdash (\tilde{x}{:}\tilde{W})^{\uparrow} P : \mathsf{Pro}[E,\tilde{W}]}$$

(OUTPUT $\uparrow$)
$$\frac{\Gamma \vdash \tilde{M} : \tilde{W} \quad \Gamma \vdash P : \mathsf{Pro}[E,\tilde{W}]}{\Gamma \vdash \langle \tilde{M} \rangle^{\uparrow} P : \mathsf{Pro}[E,\tilde{W}]}$$

The rules for input/output are not surprising. We use the notation $\Gamma \vdash \tilde{M} : \tilde{W}$ for $\Gamma \vdash M_1 : W_1, \ldots, \Gamma \vdash M_k : W_k$. In all cases, the rules require that the type of the exchanged values comply with the local exchange type of the target ambient, as expected. Interestingly, the rules for downward input/output, (INPUT M) and (OUTPUT M), do not impose any constraint on the types of the local and upward exchanges.

As we noted earlier, it would be tempting to extend the subtyping relations, and their use in the type system in several ways. Unfortunately, such extensions are unsound. We first show that subtyping between upward silent and upward non-silent processes is unsound when combined with the typing of parallel composition. To motivate, consider allowing the relation $\mathsf{Pro}[E,\mathsf{shh}] \leqslant \mathsf{Pro}[E,F]$, for $F \neq \mathsf{shh}$. Then take the ambient $a[\mathsf{in}\ b.\mathbf{0} \mid \langle M\rangle^{\uparrow}P]$ with, $M : W$ for some type W, and note that $\mathsf{in}\ b.\mathbf{0}$ can be typed as $\mathsf{Pro}[\mathsf{shh},\mathsf{shh}]$ regardless of the type of b. By the additional subtyping rule, we could then type the parallel composition as $\mathsf{Pro}[\mathsf{shh},W]$. However, if $b : \mathsf{Amb}[W',F]$ for some $W' \neq W$, the ambient a could move into b and have unsound upward exchanges after the move. By disabling subtyping on the upward component of process types, instead, $a[\mathsf{in}\ b.\mathbf{0} \mid \langle M\rangle^{\uparrow}P]$ does not type check as the types that can be deduced for the process $\mathsf{in}\ b.\mathbf{0}$ may only be of the form $\mathsf{Pro}[E,W']$ or $\mathsf{Pro}[E,\mathsf{shh}]$ for some exchange E. The same example shows that subtyping for capability types is unsound. To see the problem, note that by allowing $\mathsf{Cap}[\mathsf{shh}] \leq \mathsf{Cap}[W]$, one derives $\mathsf{in}\ b : \mathsf{Cap}[\mathsf{shh}]$, hence $\mathsf{in}\ b : \mathsf{Cap}[W']$ by subsumption, with the same problem we just explained. Finally, any form of non-trivial subtyping ambient types is clearly unsound, as ambients are read/write resources, and consequently their types must be invariant in the component exchange types.

The type system rules ensures that all process exchanges, inside and across ambient boundaries, are type correct. This follows directly from the subject reduction property stated below.

THEOREM SUBJECT REDUCTION. *If* $\Gamma \vdash P : T$ *and* $P \longrightarrow Q$*, then* $\Gamma \vdash Q : T$.

PROOF. *A corollary of Theorem 7.2.*

6. TYPED MOBILITY AND EXCHANGES – MOBILE VERSUS BOXED AMBIENTS

Having defined the type system, we now look at the impact of typing on mobility and communication, and contrast it with mobility and communication in Mobile Ambients.

We already remarked that mobility is orthogonal to the local exchanges within ambients. Thus, the types of the local exchanges of an ambient do not affect the ambient's capability to move. On the other hand, the presence of upward exchanges does enforce somewhat severe constraints over ambient mobility. Specifically, ambients with upward exchanges of type W may only traverse ambients whose local exchanges have type W.

However, when we compare the flexibility of mobility and communication in Boxed Ambients versus the corresponding constructs provided by Mobile Ambients, we find that the two calculi are essentially equivalent. We study this relationship in the rest of this section. We start by sketching a translation of BA into MA. The existence of such a translation should not come as a surprise: by allowing ambient dissolution, the open capability is, at least in principle, powerful enough to code communication across boundaries (indeed, in Section 2 we argued that this ability to dissolve boundaries is too powerful and hard to control, and we have introduced communication across boundaries to dispense with it). However, when we look at the translation more closely, we find a number of subtle problems, for which we were able to find only partial solutions. As we shall see, a more satisfactory encoding can be found for the asynchronous version of BA (cf. Section 8).

6.1 From boxed to mobile ambients

The idea of the translation draws on Gonthier's coaleshing encoding of the π-calculus from [Cardelli and Gordon 1999b]. We represent each boxed ambient with a corresponding MA ambient, and provide the latter with a local buffer: $\langle\!\langle n[P] \rangle\!\rangle = n[\mathsf{ch}[!\mathsf{open}\ \mathsf{pk}] \mid \langle\!\langle P \rangle\!\rangle]$, where $\langle\!\langle P \rangle\!\rangle$ is the translation of P. The buffer opens a packet pk which is intended to carry input or output inside the buffer itself. The names ch and pk are "well-known", i.e. they are global reserved names which are used uniformly[2] for all ambients (and the top level). This yields a translation in which the buffer ch is used to implement all the local and non-local exchanges involving n in the source term. Specifically, a local input operation within an ambient n generates an input packet that enters the buffer associated with n, reads an input after having been opened, then creates a return packed that exits the buffer and continues with the rest of the process:

$$\langle\!\langle (x)P \rangle\!\rangle_n \;=\; (\nu k)(\ \mathsf{pk}[\mathsf{in}\ \mathsf{ch}.(x)k[\mathsf{out}\ \mathsf{ch}.\langle\!\langle P \rangle\!\rangle_n]\,]\ \mid \mathsf{open}\ k\ )$$

The behavior of an output operation is captured in the same way. The idea carries over directly to the case of downward exchanges. It only requires a two-step move for the packets: first into the ambient, then into associated buffer. The case of upward communication is similar. In this case, however, we need to know the name n of the enclosing ambient.

$$\langle\!\langle (x)^{\uparrow}P \rangle\!\rangle_n \;=\; (\nu k)(\ \mathsf{pk}[\mathsf{out}\ n.\mathsf{in}\ \mathsf{ch}.(x)k[\mathsf{out}\ \mathsf{ch}.\mathsf{in}\ n.\langle\!\langle P \rangle\!\rangle_n]\,]\ \mid \mathsf{open}\ k\ )$$

$$\langle\!\langle (x)^{m}P \rangle\!\rangle_n \;=\; (\nu k)(\ \mathsf{pk}[\mathsf{in}\ m.\mathsf{in}\ \mathsf{ch}.(x)k[\mathsf{out}\ \mathsf{ch}.\mathsf{out}\ m.\langle\!\langle P \rangle\!\rangle_n]\,]\ \mid \mathsf{open}\ k\ )$$

A problem with this encoding is that the translation allows synchronizations that are not possible in the source term: specifically, the translation of $(x)^nP \mid n[n[\langle q\rangle^{\uparrow}Q]\,]$ has a reduction to the translation of $P\{x := q\} \mid n[n[Q]\,]$.

We have not been able to find encodings that solve this problem: in fact, for this and other encodings we have investigated, the inherent non-determinism of the reductions for value exchanged, combined with a synchronous semantics, appear to require a choice operator to be modeled in a satisfactory way in MA. Of course we do not exclude that a satisfactory encoding for the synchronous calculus can be found. On the other hand, the problems we outlined do suggest that the new form of communication is computationally interesting in itself, irrespective of its import on security.

6.2 From mobile to boxed ambients

Because of the presence of open in MA, a translation of Mobile Ambients in our calculus appears problematic, if at all possible. Nevertheless, we may still argue that typed communication and mobility in MA are captured with essentially no loss of expressive power in BA. To see that, it is instructive to note that the type system of Section 5 section can be specialized to only allow upward-silent ambient types of the form $\mathsf{Amb}[E,\mathsf{shh}]$, thus effectively inhibiting all forms of upward exchanges (this follows from the format of the (AMB) rule). The specialized type system provides full flexibility for mobility, while still allowing flexible forms of communication. In particular:

– *Mobility for Boxed Ambients is as flexible as* in/out*-mobility for typed Mobile Ambients.* This follows by the format of the (IN) and (OUT) rules. Capabilities exercised within upward silent ambients have type $\mathsf{Cap}[\mathsf{shh}]$, and $\mathsf{shh} \leqslant F$ for every F: consequently, upward silent ambients have full freedom of moving across ambient boundaries. Furthermore, since Boxed Ambients may not be opened, they may move regardless of the local exchanges of the ambients they traverse. As a consequence, with the specialized

[2]This uniform use of the same name is problematic in extending the translation to the typed cases. The problem is easily solved, however, by choosing names indexed by the types of their local exchanges (cf. Section 8.1).

type system, an ambient can move independently of its type, and of the type of its (intermediate and final) destinations.

– *Communication is as flexible as in the Ambient Calculus, even in the absence of upward exchanges*. "Upward silent" does not imply "non-communicating": an upward-silent ambient may very well move to a target ambient a, and communicate with it by means of downward reads and writes by a itself. Indeed, an ambient may access all of its children's anonymous channels as well as those of any incoming ambient, and all these exchanges may be of different types. In addition, the ambient may hold local exchanges of yet a different type. The encoding of channels given § 4.2 can also be used for encoding local exchanges of different types: the ambient $c[\,!(x\!:\!W)\langle x\rangle\,]$ can be viewed as a local channel c of type W, whose input output operators are $(x\!:\!W)^c$ and $\langle M\rangle^c$: the type system allows (encoded) channels of different types to be used in the same ambient.

Having illustrated the flexibility of the specialized type system, it is obvious that giving up upward exchanges is a problem: for instance, we would not be able to type-check pilot ambients, such as those used in the encoding of the channeled communications of § 4.3, whose function is to silently carry a process to a certain destination where the process eventually delivers its output to and/or receives input from its enclosing context. We solve the problem in the next section, where we study a refined type system that supports a more flexible, and type safe, integration of upward communication and mobility.

7. MODED TYPING

The design of the new type system is based on the observation that ambients enclosing upward-silent processes have no way to interfere with the local exchanges of their enclosing environments: as a consequence, such ambients may safely move across other ambients, regardless of the types of the latter. The new type system uses type modifiers to characterize the computation progress of processes and, in particular, to identify the silent and non-silent phases of the computation of the processes enclosed within ambients. The resulting typing technique, which we call *moded typing*, provides more precise information about ambient exchanges and, based on that, more flexible typings for several interesting systems, notably for the channels encoding of § 4.3 and for the encoding of the distributed language in Section 10.

7.1 Moded Types

The new type system is built around *moded* types defined by extending the structure of the types of Section 5 (henceforth *regular* types) as follows:

$$
\begin{array}{lrcl}
\textit{Expression Types} & W & ::= & \mathsf{Amb}[E,F] \mid \mathsf{Cap}[E] \mid \mathsf{Amb}^{\circ}[E,F] \\
\textit{Exchange Types} & E & ::= & \mathsf{shh} \mid W_1 \times \cdots \times W_k \\
\textit{Process Types} & T & ::= & \mathsf{Pro}[E,F] \mid \mathsf{Pro}[E,{}^{\bullet}F] \mid \mathsf{Pro}[E,{}^{\circ}F] \mid \mathsf{Pro}[E,{}^{\vartriangle}F]
\end{array}
$$

Capability types and ambient types of the form $\mathsf{Amb}[E,F]$ are exactly as in Section 5. Processes enclosed by regular ambient types have regular process types $\mathsf{Pro}[E,F]$, deduced by the same rules. On the other hand, *moded* ambient types of the form $\mathsf{Amb}^{\circ}[E,F]$ are associated with "pilot" ambients (in the sense we gave in Section 4.3), whose enclosed processes are assigned moded types, according to the following rationale:

– $\mathsf{Pro}[E, {}^{\bullet}W]$: upward silent processes with local exchanges of type E. The type W signals that processes with this type may safely be composed in parallel with processes with upward exchanges of type W.

– $\mathsf{Pro}[E, {}^{\circ}W]$: processes with local exchanges of type E and upward exchanges of type W. The upward exchanges are temporarily inactive because the process is moving.

– $\mathsf{Pro}[E, {}^{\vartriangle}W]$: processes with local exchanges of type E, that evolve into processes of type $\mathsf{Pro}[E, {}^{\circ}W]$ or $\mathsf{Pro}[E, {}^{\vartriangle}W]$ after performing upward exchanges of type W.

The syntax allows the formation of process types of the form $\mathsf{Pro}[E, {}^{\bullet}\mathsf{shh}]$, $\mathsf{Pro}[E, {}^{\circ}\mathsf{shh}]$ and $\mathsf{Pro}[E, {}^{\vartriangle}\mathsf{shh}]$: even though these types do not fit the above intuitions, and could safely be dispensed with, they are convenient in stating definitions and typing rules. The following notation is assumed throughout: μ is a metavariable that ranges over the modes $\circ, \bullet$ and $\vartriangle$, while ? is a metavariable that ranges over the set $\{\circ, \bullet, \vartriangle\}$ and the blank character. In other words, ${}^{\mu}F$ denotes any of the exchanges ${}^{\vartriangle}F, {}^{\bullet}F, {}^{\circ}F$, while ${}^{?}F$ denotes either ${}^{\mu}F$ or F. We sometimes use "_" to denote an arbitrary exchange type.

To illustrate the use of the modes associated with process types, consider the following process, where $M : W$: $(x{:}W')\langle x\rangle^m \mid \mathsf{in}\ n.\langle M\rangle^{\uparrow}\mathsf{out}\ n : \mathsf{Pro}[W', {}^{\circ}W]$. The left component of this process does not have upward exchanges. Consequently, if $m : \mathsf{Amb}[W', E]$ for some E, we can freely choose a type for the upward exchanges, and deduce $(x{:}W')\langle x\rangle^m$: $\mathsf{Pro}[W', {}^{\bullet}W]$. The right component, instead, does have upward exchanges, but is currently silent because the output prefix is blocked by the move: thus $\mathsf{in}\ n.\langle M\rangle^{\uparrow}\mathsf{out}\ n : \mathsf{Pro}[W', {}^{\circ}W]$, provided that $n : \mathsf{Amb}[W, W'']$. The type $\mathsf{Pro}[W', {}^{\circ}W]$ can also be assigned to the parallel composition, which is indeed currently silent. Interestingly, the type $\mathsf{Pro}[W', {}^{\circ}W]$ can *not* be assigned to the continuation process $\langle M\rangle^{\uparrow}\mathsf{out}\ n$ (nor to the parallel composition $(x{:}W')\langle x\rangle^m \mid \langle M\rangle^{\uparrow}\mathsf{out}\ n$), because, after consuming the capability $\mathsf{in}\ n$, the upward exchanges of this process become active: at this stage, a legal type for the process is $\mathsf{Pro}[W', {}^{\vartriangle}W]$, signaling that after the upward exchange the process enters again an upward-silent phase.

As the example shows, processes that are subject to moded typing may have different types (in fact, different modes, with the same type) at different stages of their computation. This does not break subject reduction, as it would seem. In fact, processes with moded types may be involved in a reduction only when enclosed within an ambient: the mode of the enclosed process changes according to the process' progress, but the type of the ambient itself is invariant through the reduction.

A final remark is in order to explain the role of the *moded* ambient types $\mathsf{Amb}^{\circ}[E, F]$. These types are needed to control the behavior of ambient processes enclosed within upward-silent ambients. In particular, for the system to be sound, we need to make sure that non-silent ambients never exit their parent during the upward-silent phases of the latter. Moded process types, by themselves, do not help. To see the problem, assume that an ambient, say a, is currently silent and moving across ambients with local exchanges of type, say, W. Also assume that a contains a non-silent ambient b with upward exchanges of type W' incompatible with W. As long as b is enclosed into a, its upward exchanges do not interfere with the local exchanges W of the ambients traversed by a: but if b exits a, then its upward exchanges may cause a type mismatch. The types $\mathsf{Amb}^{\circ}[E, F]$ come to the rescue, as the typing rule ensure that any ambient with such type can only be exited by ambients which have no upward exchanges.

7.2 Subtyping

The modes associated with the new class of processes induce a richer subtype structure for process types.

DEFINITION PROCESS SUBTYPING.
Let $\leqslant$ denote the relation of exchange subtyping introduced in Definition 5.1. Process subtyping is the smallest reflexive and transitive relation such that $\mathsf{Pro}[\mathsf{shh},{}^{\mu}F] \leqslant \mathsf{Pro}[E,{}^{\mu}F]$ *and in addition, satisfies the diagram on the right for all E and F.*

$$\begin{array}{ccccc} & & \mathsf{Pro}[E,{}^{\vartriangle}F] & & \\ & \nearrow & & \nwarrow & \\ \mathsf{Pro}[E,F] & & & & \mathsf{Pro}[E,{}^{\circ}F] \\ & \nwarrow & & \nearrow & \\ & & \mathsf{Pro}[E,{}^{\bullet}F] & & \end{array}$$

The intuition underlying process subtyping is as follows. As we said, the type $\mathsf{Pro}[\_,{}^{\bullet}E]$ identifies upward-silent processes that move their enclosing ambient only through locations with local exchanges of type E. Clearly, any such process can always be considered as a process of type $\mathsf{Pro}[\_,E]$ that is, as a process whose upward exchanges are of type E and that moves the enclosing ambient only through locations with local exchanges of type E. In fact, it can also be considered as a process of type $\mathsf{Pro}[\_,{}^{\circ}E]$, i.e. a temporarily upward-silent process whose upward exchanges will become active only when its enclosing ambient is in a context with local exchanges of type E. The two types $\mathsf{Pro}[\_,E]$ and $\mathsf{Pro}[\_,{}^{\circ}E]$ are incompatible, as processes of the first type may not be assumed to be (even temporarily) upward-silent, while processes of the second type may move across ambients regardless of the types of the latter (and therefore across ambients whose local exchanges are of a type different from E). On the other hand, the two types have a common super-type, i.e. the type $\mathsf{Pro}[\_,{}^{\vartriangle}E]$ which identifies processes that may be currently upward-active, and whose enclosing ambients are guaranteed to reside in contexts with local exchanges of type E.

7.3 Moded Judgments and Typing Rules

The additional expressive power of the new type system results from a more flexible typing of capabilities, which in turn is enabled by the modes associated with process types. Capabilities are typed in two modes: a "regular" mode, as in the type system of Section 5, and a "silent" mode in which some of the constraints on mobility can be lifted without consequences on safety.

The silent mode for typing capabilities is accounted for by a new form of judgment, denoted by $\Gamma \vdash\!\!\circ\, M : \mathsf{Cap}[E]$, which is useful when typing capability paths: if typed in silent mode, every intermediate move on the path may safely disregard the type of the ambient traversed along the move. The new type system includes all the typing rules from § 5, and new rules for deriving silent typings of capabilities, and moded types for processes (the complete set of rules is collected in Appendix A).

7.3.1 *Typing for Expressions.* We start with the rules for capabilities built around moded-typed ambients.

(IN $\circ$)
$$\frac{\Gamma \vdash M : \mathsf{Amb}^{\circ}[F,E] \quad F' \leqslant F}{\Gamma \vdash \mathsf{in}\ M : \mathsf{Cap}[F']}$$

(OUT $\circ$)
$$\frac{\Gamma \vdash M : \mathsf{Amb}^{\circ}[E,F]}{\Gamma \vdash \mathsf{out}\ M : \mathsf{Cap}[\mathsf{shh}]}$$

The (IN $\circ$) rule has the exact same format as the corresponding (IN) rule of Section 5. Instead, for the reasons we explained earlier, exiting a pilot ambient requires the exiting ambient to be upward silent. The next rules derive moded judgments for capabilities.

$$\text{(POLYCAP)}\quad \frac{\Gamma \vdash M : \mathsf{Cap}[E]}{\Gamma \vdash_{\circ} M : \mathsf{Cap}[E]} \qquad \text{(POLYPATH)}\quad \frac{\Gamma \vdash_{\circ} M_1 : \mathsf{Cap}[E_1] \quad \Gamma \vdash_{\circ} M_2 : \mathsf{Cap}[E_2]}{\Gamma \vdash_{\circ} M_1.M_2 : \mathsf{Cap}[E_2]}$$

By (POLYCAP), well-typed capabilities type check also under silent typing. In addition, capability paths can be typed with more flexibility in silent mode. According to the (POLYPATH) rule, one may disregard the exchange types of the ambients traversed at intermediate steps on the path (as no exchanges take place during those steps) and only needs to trace precise information on the last move on the path. This effectively corresponds to interpreting $\mathsf{Cap}[E]$ as the type of capability paths whose *last* move requires upward exchanges of type E. The silent typing of capabilities we just illustrated is used in conjunction with the typing of processes in prefix form, to derive moded types as we discuss next.

7.3.2 *Typing of Processes.* "Regular" types for processes are deduced by the same typing rules introduced in § 5. Moded process types are derived by new rules: we start with the rules for prefixed processes of the form $M.P$.

$$\text{(PREFIX }\circ\text{)}\quad \frac{\Gamma \vdash_{\circ} M : \mathsf{Cap}[G] \quad \Gamma \vdash P : \mathsf{Pro}[E, {}^{\circ}F]}{\Gamma \vdash M.P : \mathsf{Pro}[E, {}^{\circ}F]} \qquad \text{(PREFIX }\triangle\text{)}\quad \frac{\Gamma \vdash_{\circ} M : \mathsf{Cap}[F] \quad \Gamma \vdash P : \mathsf{Pro}[E, {}^{\triangle}F]}{\Gamma \vdash M.P : \mathsf{Pro}[E, {}^{\circ}F]}$$

The (PREFIX $\circ$) rule is one of the cornerstones of the moded typing system. It lifts the restriction, distinctive of the (PREFIX) rule of Section 5, that the exchange type G in the type of M must be compatible with the type F of the upward exchanges of P. As a consequence, M may be exercised irrespective of the type F. This is only sound if P has a $\circ$-moded type, for in that case P is itself a prefix, and hence upward silent when M is exercised. On the other hand, in (PREFIX $\triangle$) P may be have active upward exchanges, and thus the rule imposes the same constraints as the regular (PREFIX) rule, by requiring the upward exchanges of M and P to be consistent (equal). In other words, the *last* move of the prefix must be compatible with the upward exchanges that the process may have right after. Notice also that, by subsumption, (PREFIX $\triangle$) assigns moving types to processes of the form $M.P$ with P of type $\mathsf{Pro}[E, F]$.

$$\text{(PREFIX }\bullet\text{)}\quad \frac{\Gamma \vdash M : \mathsf{Cap}[F] \quad \Gamma \vdash P : \mathsf{Pro}[E, {}^{\bullet}F]}{\Gamma \vdash M.P : \mathsf{Pro}[E, {}^{\bullet}F]}$$

The rule (PREFIX $\bullet$) types silent processes running in a context whose upward exchanges (if any) have type F. In this case, the regular typing of the path M guarantees that P is type compatible with the local exchanges of the ambients hit on the move.

The next two rules apply to parallel compositions: two rules, and an appeal to subsumption, capture all cases.

$$\text{(PARALLEL } \mu \text{ LEFT)}\quad \frac{\Gamma \vdash P : \mathsf{Pro}[E, {}^{\mu}F] \quad \Gamma \vdash Q : \mathsf{Pro}[E, {}^{\bullet}F]}{\Gamma \vdash P \mid Q : \mathsf{Pro}[E, {}^{\mu}F]}$$

$$\text{(PARALLEL } \mu \text{ RIGHT)}\quad \frac{\Gamma \vdash P : \mathsf{Pro}[E, {}^{\bullet}F] \quad \Gamma \vdash Q : \mathsf{Pro}[E, {}^{\mu}F]}{\Gamma \vdash P \mid Q : \mathsf{Pro}[E, {}^{\mu}F]}$$

If P and Q are upward-silent (i.e. with upward exchanges ${}^{\bullet}F$), then $P \mid Q$ is also upward silent (with upward exchanges ${}^{\bullet}F$). $P \mid Q$ can be typed as moving (that is, with upward exchanges ${}^{\circ}F$), only when (i) either one of P or Q is moving and (ii) the other process is upward silent and type compatible with the exchanges of the moving process. The same reasoning applies to the other modes. Two rules are needed to handle the two cases when the moving subprocess is P or Q.

The rules for the inactive process and processes built from restrictions present no surprise. This is not true of replicated processes: given the congruence law $!P \equiv P \mid !P$, the reasoning we just made about parallel composition implies that the only mode derivable for a replicated process $!P$ is the silent mode, provided that P is also silent. Consequently, the only two possible types for a replicated process are a "regular" type (deduced by the (REPLICATION) rule of Section 5) or a silent type, derived as follows:

$$\text{(REPL } \bullet\text{)}\quad \frac{\Gamma \vdash P : \mathsf{Pro}[E, {}^{\bullet}F]}{\Gamma \vdash\ !P : \mathsf{Pro}[E, {}^{\bullet}F]}$$

For processes in ambient form we need new rules. The rule (AMB) from § 5 is modified (see Appendix A) to deduce upward-silent types, compatible with all the other modes. Two new rules handle the cases processes enclosed in pilot ambients, depending on whether such processes are moving or not.

$$\text{(AMB } \vartriangle\text{)}\quad \frac{\Gamma \vdash M : \mathsf{Amb}^{\circ}[E,F] \quad \Gamma \vdash P : \mathsf{Pro}[E, {}^{\vartriangle}F]}{\Gamma \vdash M[P] : \mathsf{Pro}[F, {}^{\bullet}H]}$$

$$\text{(AMB } \circ\text{)}\quad \frac{\Gamma \vdash M : \mathsf{Amb}^{\circ}[E,F] \quad \Gamma \vdash P : \mathsf{Pro}[E, {}^{\circ}F]}{\Gamma \vdash M[P] : \mathsf{Pro}[G, {}^{\bullet}H]}$$

In (AMB $\vartriangle$) P is not moving, and the rule imposes type constraints equivalent to those imposed by the (AMB) rule: this is needed for soundness, as the judgment $\Gamma \vdash P : \mathsf{Pro}[E, {}^{\vartriangle}F]$ could be derived by subsumption from $\Gamma \vdash P : \mathsf{Pro}[E,F]$. Instead, if P is moving, as in (AMB $\circ$), its upward exchanges are blocked by the move, and therefore the type of the local exchanges of the process $M[P]$ can be chosen arbitrarily, as the unrestrained G in the conclusion indicates.

We conclude with the rules for input-output.

$$\text{(INPUT } \star\, \mu\text{)}\quad \frac{\Gamma, \tilde{x}{:}\tilde{W} \vdash P : \mathsf{Pro}[\tilde{W}, {}^{\mu}F]}{\Gamma \vdash (\tilde{x}{:}\tilde{W})P : \mathsf{Pro}[\tilde{W}, {}^{\mu}F]}$$

$$\text{(OUTPUT } \star\, \mu\text{)}\quad \frac{\Gamma \vdash \tilde{M} : \tilde{W} \quad \Gamma \vdash P : \mathsf{Pro}[\tilde{W}, {}^{\mu}F]}{\Gamma \vdash \langle \tilde{M} \rangle P : \mathsf{Pro}[\tilde{W}, {}^{\mu}F]}$$

$$\text{(INPUT } \uparrow \vartriangle\text{)}\quad \frac{\Gamma, \tilde{x}{:}\tilde{W} \vdash P : \mathsf{Pro}[F, {}^{\vartriangle}\tilde{W}]}{\Gamma \vdash (\tilde{x}{:}\tilde{W})^{\uparrow} P : \mathsf{Pro}[F, {}^{\vartriangle}\tilde{W}]}$$

$$\text{(OUTPUT } \uparrow \vartriangle\text{)}\quad \frac{\Gamma \vdash \tilde{M} : \tilde{W} \quad \Gamma \vdash P : \mathsf{Pro}[F, {}^{\vartriangle}\tilde{W}]}{\Gamma \vdash \langle \tilde{M} \rangle^{\uparrow} P : \mathsf{Pro}[F, {}^{\vartriangle}\tilde{W}]}$$

Local communications are not affected by modes: it is the mode of the continuation process that determines the moded type of the input/output process itself. A process that starts with an upward exchange has a non-moving type, for obvious reasons, and its type depends on the type of the continuation. If the continuation is of type $\mathsf{Pro}[F, {}^{\bullet}W]$ or $\mathsf{Pro}[F,W]$, then the process —which is clearly not silent— can be typed as $\mathsf{Pro}[F,W]$. These cases are captured by the rule (INPUT/OUTPUT $\uparrow$) of § 5 (together with subsumption for the case $\mathsf{Pro}[F, {}^{\bullet}W]$). If instead the continuation has type $\mathsf{Pro}[F, {}^{\vartriangle}W]$ or $\mathsf{Pro}[F, {}^{\circ}W]$, as in (INPUT/OUTPUT $\uparrow \vartriangle$), we can just say that the process may eventually evolve into a moving process, hence the type $\mathsf{Pro}[F, {}^{\vartriangle}W]$ in the conclusion.

Finally, downward communications are not affected by whether the target ambient is moded or not. The rules from § 5 work just as well for the new system: two new rules, with the same format, handle the case when target ambient is moded:

(INPUT $M \circ$)

$$\frac{\Gamma \vdash M : \mathsf{Amb}^{\circ}[\tilde{W}, E] \quad \Gamma, \tilde{x}{:}\tilde{W} \vdash P : T}{\Gamma \vdash (\tilde{x}{:}\tilde{W})^{M} P : T}$$

(OUTPUT $M \circ$)

$$\frac{\Gamma \vdash N : \mathsf{Amb}^{\circ}[\tilde{W}, E] \quad \Gamma \vdash \tilde{M} : \tilde{W} \quad \Gamma \vdash P : T}{\Gamma \vdash \langle \tilde{M} \rangle^{N} P : T}$$

As a final remark, note that in all the output rules, the typing of the expression M being output is subject to "regular" typing. As a consequence, capability paths may be communicated only if well-typed under regular typing. This restriction could be lifted, had we employed capability types with modes, instead of typing capabilities with moded judgements. In fact, adding capability types with modes would indeed result into a slightly more expressive system (one which would allow "moded" paths to be communicated). On the other hand, the current solution has the advantage of requiring minimal changes to the syntax of expression types, those occurring in the typed syntax, and for this reason it had our preference.

7.4 Subject Reduction

Subject reduction for the new system is proved following the standard technique: a detailed proofs is in Appendix B.

THEOREM SUBJECT REDUCTION. *If* $\Gamma \vdash P : T$ *and* $P \longrightarrow Q$ *then* $\Gamma \vdash Q : T$.

We illustrate the moded-type system at work on two examples. First consider the following process, where we use the primitive types *int* and *bool* for convenience:

$$(x : int)P \mid b[\,\langle 5 \rangle^{\uparrow}\, \mathsf{in}\ a\,] \mid a[\,(x : bool)Q\,]$$

This process is clearly type safe (insofar as P and Q are safe), as the upward exchange in b is compatible with the local exchanges of type *int* occurring at top level. Once the exchange is consumed, b becomes upward silent, and may safely move into a, even though a has *bool* exchanges. With the the system of Section 5, the process is ill-typed. To see that, first observe that the ambient names a and b may only be assigned the types $\mathsf{Amb}[bool, \_]$ and $\mathsf{Amb}[\_, int]$. Then, for the process to type-check, we would need to derive the typing $\langle 5 \rangle^{\uparrow}\mathsf{in}\ a : \mathsf{Pro}[\_, int]$. This fails because the move into a violates the type constraint that requires *int* to be compatible with the local exchanges of a.

Instead, with the moded type system one can derive $\langle 5 \rangle^{\uparrow}\mathsf{in}\ a : \mathsf{Pro}[\_, {}^{\circ}int]$, provided that b is assigned a pilot ambient type. Below we give a type derivation for $b[\,\langle 5 \rangle^{\uparrow}\mathsf{in}\ a\,]$, assuming that Γ is the type environment $a : \mathsf{Amb}[bool, \_], b : \mathsf{Amb}^{\circ}[\_, int]$.

$$
\dfrac{\Gamma \vdash b : \mathsf{Amb}^{\circ}[\_, \mathit{int}] \qquad \dfrac{\Gamma \vdash 5 : \mathit{int} \qquad \dfrac{\dfrac{\dfrac{\Gamma \vdash a : \mathsf{Amb}[\mathit{bool}, \_]}{\Gamma \vdash \mathsf{in}\ a : \mathsf{Cap}[\mathit{bool}]} \qquad \Gamma \vdash \mathbf{0} : \mathsf{Pro}[\_, {}^{\circ}\mathit{int}]}{\Gamma \vdash \mathsf{in}\ a.\mathbf{0} : \mathsf{Pro}[\_, {}^{\circ}\mathit{int}]}\ (\text{PREFIX} \circ)}{\Gamma \vdash \mathsf{in}\ a.\mathbf{0} : \mathsf{Pro}[\_, {}^{\vartriangle}\mathit{int}]}\ (\text{SUBSUMPTION})}{\Gamma \vdash \langle 5 \rangle^{\uparrow} \mathsf{in}\ a.\mathbf{0} : \mathsf{Pro}[\_, {}^{\vartriangle}\mathit{int}]}\ (\text{OUTPUT} \uparrow \vartriangle)}{\Gamma \vdash b[\,\langle 5 \rangle^{\uparrow} \mathsf{in}\ a\,] : \mathsf{Pro}[\mathit{int}, \_]}\ (\text{Amb} \vartriangle)
$$

Notice, further, that after the upward exchange and the move into a, the residual process $b[\,]$ can be given any type, in particular the type $\mathsf{Pro}[\mathit{bool}, \_]$ needed to type-check the occurrence of this process within a.

As second example, we look at the typed version of the channel encoding of Section 4.3. In the typed case, the encoding is defined as follows (see Section 4.3):

$$
\begin{array}{lcl}
\langle\!\langle c^{m}\langle x \rangle \rangle\!\rangle_{n} & = & (\nu p{:}\mathsf{Amb}^{\circ}[\mathsf{shh}, W])p[\,\mathsf{in}\ m.\mathsf{in}\ c.\langle x \rangle^{\uparrow}\,] \\
\langle\!\langle c^{\uparrow}\langle x \rangle \rangle\!\rangle_{n} & = & (\nu p{:}\mathsf{Amb}^{\circ}[\mathsf{shh}, W])p[\,\mathsf{out}\ n.\mathsf{in}\ c.\langle x \rangle^{\uparrow}\,] \\
\langle\!\langle c^{m}(x{:}W).P \rangle\!\rangle_{n} & = & (\nu p{:}\mathsf{Amb}^{\circ}[W, W])p[\,\mathsf{in}\ m.\mathsf{in}\ c.(x{:}W)^{\uparrow}\mathsf{out}\ c.\mathsf{out}\ m.\langle x \rangle\,] \mid (x{:}W)^{p} \langle\!\langle P \rangle\!\rangle_{n} \\
\langle\!\langle c^{\uparrow}(x{:}W).P \rangle\!\rangle_{n} & = & (\nu p{:}\mathsf{Amb}^{\circ}[W, W])p[\,\mathsf{out}\ n.\mathsf{in}\ c.(x{:}W)^{\uparrow}\mathsf{out}\ c.\mathsf{in}\ n.\langle x \rangle\,] \mid (x{:}W)^{p} \langle\!\langle P \rangle\!\rangle_{n}
\end{array}
$$

Remarkably, the definition is independent of the types of the two ambients m and n traversed by the ambient p: this flexibility is enabled by the typing of p as a pilot ambient. We give a type derivation for the case of downward input as representative. With no loss of generality, we make the following assumptions: $\Gamma \vdash \langle\!\langle P \rangle\!\rangle_{n}{:}\mathsf{Pro}[E, {}^{?}F]$, with Γ a type environment in which $m{:}\mathsf{Amb}^{?}[G, H], c{:}\mathsf{Amb}[W, \mathsf{shh}]$ and $p{:}\mathsf{Amb}^{\circ}[W, W]$, and E, F, G and H are arbitrary exchange types. We reconstruct a type derivation for the judgment:

$$\Gamma \vdash p[\,\mathsf{in}\ m.\mathsf{in}\ c.(x{:}W)^{\uparrow}\mathsf{out}\ c.\mathsf{out}\ m.\langle x \rangle\,] : \mathsf{Pro}[E, {}^{\bullet}F].$$

This judgment is derived by (AMB $\circ$), provided that the process enclosed in p can be typed with mode $\circ$, i.e. if

$$\Gamma \vdash \mathsf{in}\ m.\mathsf{in}\ c.(x{:}W)^{\uparrow}\mathsf{out}\ c.\mathsf{out}\ m.\langle x \rangle : \mathsf{Pro}[W, {}^{\circ}W].$$

This follows by (PREFIX $\circ$) from $\Gamma \vdash^{\circ} \mathsf{in}\ m{:}\mathsf{Cap}[G]$ and

$$\Gamma \vdash \mathsf{in}\ c.(x{:}W)^{\uparrow}\mathsf{out}\ c.\mathsf{out}\ m.\langle x \rangle : \mathsf{Pro}[W, {}^{\circ}W].$$

G is the type of the local exchanges in m, and moded typing allows G to be any type, irrespective of W (notice that this is would not be true without moded types as the judgement $\Gamma \vdash \mathsf{in}\ c.(x{:}W)^{\uparrow}\mathsf{out}\ c.\mathsf{out}\ m.\langle x \rangle : \mathsf{Pro}[W, W]$ is derivable only if $G \leqslant W$). The last judgment follows by (PREFIX $\vartriangle$) from $\Gamma \vdash^{\circ} \mathsf{in}\ c{:}\mathsf{Cap}[W]$ and from

$$\Gamma \vdash (x{:}W)^{\uparrow}\mathsf{out}\ c.\mathsf{out}\ m.\langle x \rangle : \mathsf{Pro}[W, {}^{\vartriangle}W].$$

This judgment can be derived by (INPUT $\uparrow \vartriangle$) from

$$\Gamma, x{:}W \vdash \mathsf{out}\ c.\mathsf{out}\ m.\langle x \rangle{:}\mathsf{Pro}[W, {}^{\vartriangle}W].$$

Again, we rely on moded typing: the whole process type-checks since the move that precedes the upward output routes the ambient into an environment with the right exchange type. Deriving the last judgment is not difficult. From $\Gamma, x{:}W \vdash^{\circ} \mathsf{out}\ m{:}\mathsf{Cap}[H]$ and from $\Gamma, x{:}W \vdash \langle x\rangle{:}\mathsf{Pro}[W, {}^{\circ}W]$, we have $\Gamma, x{:}W \vdash \mathsf{out}\ m.\langle x\rangle : \mathsf{Pro}[W, {}^{\circ}W]$. Again, moded types are required here, as $\Gamma, x{:}W \vdash \mathsf{out}\ m.\langle x\rangle : \mathsf{Pro}[W, W]$ is not derivable.

Now, from the last judgment and from $\Gamma, x{:}W \vdash^{\circ} \mathsf{out}\ c{:}\mathsf{Cap}[\mathsf{shh}]$ an application of (PREFIX $\circ$) yields $\Gamma, x{:}W \vdash \mathsf{out}\ c.\mathsf{out}\ m.\langle x\rangle{:}\mathsf{Pro}[W, {}^{\circ}W]$ as desired. To conclude, we obtain the desired typing from $\mathsf{Pro}[W, {}^{\circ}W] \leqslant \mathsf{Pro}[W, {}^{\vartriangle}W]$, by subsumption and (INPUT $\uparrow \vartriangle$).

8. ASYNCHRONOUS BOXED AMBIENTS

A calculus for distributed computation cannot rely on synchronous *rendez-vous* as the only mechanism for process interaction and value exchange. Indeed, the fundamental role of asynchronous primitives in distributed systems is well-understood (cf. [Fournet et al. 1996; Cardelli 1999]), and motivated by widely agreed design principles and practical experience with implementation [Bryce and Vitek 2001; Fournet et al. 2000].

As we noted in Section 3, asynchronous exchanges can be recovered in BA by reconsidering the semantics of the output process forms. In particular, we suggested two solutions for making the output $\langle M\rangle^{\eta}P$ asynchronous: either code it by using continuation-less outputs (so that the asynchronous output $\langle M\rangle^{\eta}P$ is encoded by the parallel composition $\langle M\rangle^{\eta} \mid P$), or introduce the structural law $\langle M\rangle^{\eta}P \equiv \langle M\rangle^{\eta} \mid P$. The first solution allows synchronous and asynchronous output to coexist, the second solution yields a purely asynchronous calculus.

Both the alternatives have a fundamental problem, namely that splitting an output form into a parallel composition has the effect of essentially defeating moded typing. Moded typing is possible, and effective, only along a single thread, while the coding of asynchronous output introduces parallel compositions and leaves no residual following an output. In particular, with $\eta = \uparrow$, $\langle M\rangle^{\eta}P$ and $\langle M\rangle^{\eta} \mid P$ are only equivalent under the type system of § 5, not with moded types (thus subject congruence fails with $\langle M\rangle^{\uparrow}P \equiv \langle M\rangle^{\uparrow} \mid P$ and moded typing).

To see the problem, consider the process $(x : int)P \mid b[\![\, \langle 5\rangle^{\uparrow}\mathsf{in}\ a \,]\!] \mid a[\![\, (x : bool)Q \,]\!]$, whose typing we studied in Section 7. If we take $\langle 5\rangle^{\uparrow}\mathsf{in}\ a$ and transform it as suggested above, the process does not type check, as the typing of $b[\![\, \langle 5\rangle^{\uparrow} \mid \mathsf{in}\ a \,]\!]$ is not derivable, even under the assumption that $b : \mathsf{Amb}^{\circ}[\_, int]$. The problem is with the rules for parallel composition, which require the following typing to be derivable: $\mathsf{in}\ a.\mathbf{0} : \mathsf{Pro}[\_, {}^{\bullet}int]$. This fails due to the format of the (PREFIX $\bullet$) rule, which has the same requirements as (PREFIX).

Fortunately, however, the problem is not a consequence of moded typing and asynchrony being inherently incompatible. To see that, observe that in $\langle M\rangle^{\uparrow}P$ the continuation P could be typed with a mode independently of whether the prefix denotes synchronous or asynchronous output. All that matters for P to receive a sound "moving" type is that $\langle M\rangle$ gets delivered to the parent ambient before unleashing P: once delivered, whether or not $\langle M\rangle$ also synchronizes with local input is irrelevant. Based on this observation, asynchronous output and moded typing can be reconciled by resorting to a more careful definition of the congruence law, namely, one in which $\langle M\rangle^{\eta}P \equiv \langle M\rangle^{\eta} \mid P$ only if $\eta \neq \uparrow$. Instead, when $\eta = \uparrow$, we re-state the congruence law as a reduction and make it location-aware so that the output is delivered to the appropriate ambient: $n[\![\, \langle M\rangle^{\uparrow}P \mid Q \,]\!] \longrightarrow \langle M\rangle \mid n[\![\, P \mid Q \,]\!]$.

With this reduction, the problem with moded types is solved: an upward output followed by a move, such as $\langle N\rangle^{\uparrow}\mathsf{in}\ n.P$ may safely be typed with mode $\vartriangle$ (based on the mode $\circ$ for the type of $\mathsf{in}\ n.P$) regardless of whether the output synchronizes or not.

We can now define the asynchronous calculus formally. The (typed) syntax is unchanged, as one can still interpret[3] $\langle M\rangle^\eta$ as the process $\langle M\rangle^\eta\mathbf{0}$. Instead, the semantics is different, as input and output prefixes synchronize only when the latter have null continuation. Table IV summarizes the new top-level reductions, and the new congruence laws for asynchronous communication.

Table IV Asynchronous reductions

Structural Congruence

(Struct Output)	$\langle\tilde M\rangle^\eta P \;\equiv\; \langle\tilde M\rangle^\eta \mid P$	$(\eta \neq \uparrow)$

Communication

(local)	$(\tilde x:\tilde W)P \mid \langle\tilde M\rangle \;\longrightarrow\; P\{\tilde x := \tilde M\}$
(input n)	$(\tilde x:\tilde W)^n P \mid n[\langle\tilde M\rangle \mid Q] \;\longrightarrow\; P\{\tilde x := \tilde M\} \mid n[Q]$
(input $\uparrow$)	$n[(\tilde x:\tilde W)^\uparrow P \mid Q] \mid \langle\tilde M\rangle \;\longrightarrow\; n[P\{\tilde x := \tilde M\} \mid Q]$
(output n)	$\langle\tilde M\rangle^n \mid n[P] \;\longrightarrow\; n[\langle\tilde M\rangle \mid P]$
(output $\uparrow$)	$n[\langle\tilde M\rangle^\uparrow P \mid Q] \;\longrightarrow\; \langle\tilde M\rangle \mid n[P \mid Q]$

The relation of asynchronous reduction is obtained by replacing the communication reduction of Table III with the corresponding reductions in Table IV. Notice that the semantics of the output form $\langle M\rangle^\eta P$ is now truly asynchronous, as P is unleashed regardless of whether there is a matching input process. To illustrate, consider again our running example: with the asynchronous semantics, one has the following sequence of reductions

$$(x:int)P \mid b[\langle 5\rangle^\uparrow \mathsf{in}\ a] \mid a[(x:bool)Q] \Longrightarrow (x:int)P \mid \langle 5\rangle \mid a[b[\,] \mid (x:bool)Q]$$

Interestingly, the new reductions are compatible, and sound, with the moded typing system. Type soundness follows directly from the following two results (proved in Appendix B).

LEMMA SUBJECT CONGRUENCE. *Assume $\eta \neq \uparrow$. Then, $\Gamma \vdash \langle M\rangle^\eta P : T$ if and only if $\Gamma \vdash \langle M\rangle^\eta \mid P : T$, where both judgements are derived in the system of Section 7.*

THEOREM SUBJECT REDUCTION IN THE ASYNCHRONOUS CALCULUS. *Let $\Gamma \vdash P : T$ with the system of Section 7, and $P \longrightarrow Q$ with the asynchronous reduction rules. Then $\Gamma \vdash Q : T$.*

8.1 Asynchronous Boxed Ambient vs Mobile Ambients

We mentioned in Section 6 that the asynchronous semantics enables the definition of more robust translation of Boxed Ambients in MA. The translation is defined formally in in Table V. To ease the presentation, we restrict to a monadic version of BA in which only names (and not capabilities) can be exchanged. There is no fundamental difficulty in extending

[3]Indeed, although operationally equivalent, the two terms could be distinguished to reflect their respective nature, namely: $\langle M\rangle^\eta\mathbf{0}$ indicates an output operation with null continuation, while $\langle M\rangle^\eta$ denotes a piece of data which has been delivered to η (when $\eta \neq \star$), or a one-place buffer , when $\eta = \star$.

the definition to the general case (some care is required to handle the typed exchange of capabilities).

The translation is typed: with respect to the untyped case, the main difference is in the use of a family of names ch_W and pk, indexed on types, with the implicit assumption that $\mathsf{ch}_W, \mathsf{pk}_W : \mathsf{Amb}[W]$ for all type W. This indexing is required for typing, as each pair of names enables exchanges of the corresponding type.

Table V Encoding of BA into MA

Types:

$$\begin{aligned}
\langle\!\langle \mathsf{Amb}[W,E] \rangle\!\rangle &= \mathsf{Amb}[\langle\!\langle W \rangle\!\rangle],\ \langle\!\langle \mathsf{Amb}[\mathsf{shh},E] \rangle\!\rangle = \mathsf{Amb}[\mathsf{shh}] \\
\langle\!\langle \mathsf{Pro}[W,E] \rangle\!\rangle &= \mathsf{Pro}[\langle\!\langle W \rangle\!\rangle],\ \langle\!\langle \mathsf{Pro}[\mathsf{shh},E] \rangle\!\rangle = \mathsf{shh}
\end{aligned}$$

Type environments: $\langle\!\langle x_1 : W_1, \ldots, x_n : W_n \rangle\!\rangle = x_1 : \langle\!\langle W_1 \rangle\!\rangle, \ldots, x_n : \langle\!\langle W_n \rangle\!\rangle$

Terms: assume $\Gamma \vdash n : \mathsf{Amb}[W,W']$ and $\Gamma \vdash m : \mathsf{Amb}[W'',E]$

$$\begin{aligned}
\langle\!\langle \Gamma \triangleright \langle q \rangle P \rangle\!\rangle_n &= (\nu k : \mathsf{Amb}[\langle\!\langle W \rangle\!\rangle])(\mathsf{open}\ k \mid \\
&\qquad \mathsf{pk}_W[\langle q \rangle \mid \mathsf{in}\ \mathsf{ch}_W.k[\mathsf{out}\ \mathsf{ch}_W.\langle\!\langle \Gamma \triangleright P \rangle\!\rangle_n]]) \\
\langle\!\langle \Gamma \triangleright (x : W)P \rangle\!\rangle_n &= (\nu k : \mathsf{Amb}[\langle\!\langle W \rangle\!\rangle])(\mathsf{open}\ k \mid \\
&\qquad \mathsf{pk}_W[\mathsf{in}\ \mathsf{ch}_W.(x{:}\langle\!\langle W \rangle\!\rangle)k[\mathsf{out}\ \mathsf{ch}_W.\langle\!\langle \Gamma \triangleright P \rangle\!\rangle_n]]) \\
\langle\!\langle \Gamma \triangleright \langle q \rangle^m P \rangle\!\rangle_n &= (\nu k : \mathsf{Amb}[\langle\!\langle W \rangle\!\rangle])(\mathsf{open}\ k \mid \\
&\qquad \mathsf{pk}_{W''}[\langle q \rangle \mid \mathsf{in}\ m.\mathsf{in}\ \mathsf{ch}_{W''}.k[\mathsf{out}\ \mathsf{ch}_{W''}.\mathsf{out}\ m\langle\!\langle \Gamma \triangleright P \rangle\!\rangle_n]]) \\
\langle\!\langle \Gamma \triangleright (x : W'')^m P \rangle\!\rangle_n &= (\nu k : \mathsf{Amb}[\langle\!\langle W \rangle\!\rangle])(\mathsf{open}\ k \mid \\
&\qquad \mathsf{pk}_{W''}[\mathsf{in}\ m.\mathsf{in}\ \mathsf{ch}_{W''}.(x{:}\langle\!\langle W'' \rangle\!\rangle)k[\mathsf{out}\ \mathsf{ch}_{W''}.\mathsf{out}\ m\langle\!\langle \Gamma \triangleright P \rangle\!\rangle_n]]) \\
\langle\!\langle \Gamma \triangleright \langle q \rangle^{\uparrow} P \rangle\!\rangle_n &= (\nu k : \mathsf{Amb}[\langle\!\langle W \rangle\!\rangle])(\mathsf{open}\ k \mid \\
&\qquad \mathsf{pk}_{W'}[\langle q \rangle \mid \mathsf{out}\ n.\mathsf{in}\ \mathsf{ch}_{W'}.k[\mathsf{out}\ \mathsf{ch}_{W'}.\mathsf{in}\ n.\langle\!\langle \Gamma \triangleright P \rangle\!\rangle_n]]) \\
\langle\!\langle \Gamma \triangleright (x : W')^{\uparrow} P \rangle\!\rangle_n &= (\nu k : \mathsf{Amb}[\langle\!\langle W \rangle\!\rangle])(\mathsf{open}\ k \mid \\
&\qquad \mathsf{pk}_{W'}[\mathsf{out}\ n.\mathsf{in}\ \mathsf{ch}_{W'}.(x{:}\langle\!\langle W' \rangle\!\rangle)k[\mathsf{out}\ \mathsf{ch}_{W'}.\mathsf{in}\ n.\langle\!\langle \Gamma \triangleright P \rangle\!\rangle_n]]) \\
\langle\!\langle \Gamma \triangleright m[P] \rangle\!\rangle_n &= m[\mathsf{ch}_{W''}[!\mathsf{open}\ \mathsf{pk}_{W''}] \mid \langle\!\langle \Gamma \triangleright P \rangle\!\rangle_m] \\
\langle\!\langle \Gamma \triangleright P \mid Q \rangle\!\rangle_n &= \langle\!\langle \Gamma \triangleright P \rangle\!\rangle_n \mid \langle\!\langle \Gamma \triangleright Q \rangle\!\rangle_n \\
\langle\!\langle \Gamma \triangleright (\nu m : W)P \rangle\!\rangle_n &= (\nu m : \langle\!\langle W \rangle\!\rangle)\langle\!\langle \Gamma, m : W \triangleright P \rangle\!\rangle_n \\
\langle\!\langle \Gamma \triangleright M.P \rangle\!\rangle_n &= M.\langle\!\langle \Gamma \triangleright P \rangle\!\rangle_n \\
\langle\!\langle \Gamma \triangleright !P \rangle\!\rangle_n &= !\langle\!\langle \Gamma \triangleright P \rangle\!\rangle_n
\end{aligned}$$

The translation has interesting properties. It is type preserving, namely: if $\Gamma \vdash P : \mathsf{Pro}[E,F]$ in BA, then one can show that $\langle\!\langle \Gamma \rangle\!\rangle \vdash \langle\!\langle \Gamma \triangleright P \rangle\!\rangle : \langle\!\langle \mathsf{Pro}[E,F] \rangle\!\rangle$. Also, the translation simulates the reductions of the source (asynchronous) calculus correctly. Unfortunately, there are still two remaining problems. First the translation is not fully compositional, as the complete encoding needs a further step, to add a buffer to the top level: $\langle\!\langle \Gamma \triangleright P \rangle\!\rangle = \langle\!\langle \Gamma \triangleright P \rangle\!\rangle_{\mathsf{top}} \mid \mathsf{ch}_W[\,]$ where W is the type of the local exchanges of P. Secondly, the protocols that implement the exchanges across boundaries are not atomic, and hence subject to interferences. To make them atomic, one would need further hypotheses of the source term, akin to those encompassed by the notion of *single-threadedness* defined for Safe Ambients [Levi and Sangiorgi 2000]. For instance, for the upward exchanges to be implemented correctly, we would need to assume that the ambient n exited at the start of the protocol does not move until the ambient k is back into n (hence the protocol is complete).

8.2 Synchrony versus asynchrony

The choice of synchronous versus asynchronous communication has various consequences on the calculus, specifically, on the security guarantees that can be made for it.

On one side, it is well known that synchronous output generates hard-to-detect flow of information based on synchronization. For example, with the synchronous semantics, in the system $a[\,Q \mid b[\,\langle M\rangle P\,]\,]$, the sub-ambient b gets to know exactly when (and if) Q makes a downward read to its contents, thus causing an implicit flow of information from the reader to the writer: this makes non-interference [Goguen and Meseguer 1982; Focardi and Gorrieri 1997] hard to satisfy.

On the other hand, by adopting asynchronous output we effectively give up *mediation* (see § 3), that is, control over the interactions between sibling ambients. With the synchronous semantics, no ambient can be "spoiled" with unexpected (and possibly unwanted) output by its enclosing or enclosed ambients. As an example, consider the system $a[\,(x{:}W)^b P \mid b[\,c[\,\langle M\rangle^{\uparrow} \mid Q\,]\,]\,]$ which type-checks provided that $M : W$ and the ambient b has type $\mathsf{Amb}[W,F]$ for some F. With the synchronous reductions there is no way for the upward output in c and the downward input in a to synchronize.

Instead, in the asynchronous case, the initial configuration evolves into the process $a[\,(x{:}W)^b P \mid b[\,\langle M\rangle \mid c[\,Q\,]\,]\,]$, and by a further reduction the ambient a gets hold of the message $\langle M\rangle$ without any mediation by b. Similarly, two siblings may establish a hidden channel, as $b[\,a[\,(x{:}W)^{\uparrow} P\,] \mid c[\,\langle M\rangle^{\uparrow} Q\,]\,]$ reduces in two steps to $b[\,a[\,P\{x := M\}\,] \mid c[\,Q\,]\,]$. Both situations result in security breaches, based on the presence of hidden channels, that cannot be prevented by the primitives of the calculus, as it stands. Fortunately, however, one can resort to types and type analysis to provide stronger security guarantees, and enhanced policies for access control. We discuss this aspect in the next section.

9. ACCESS CONTROL BY TYPING

The access control framework we address is an instance of the standard Mandatory Access Control policies in multi-level security environments [Bell and Padula 1976; Gollmann 1999]. The domain of security levels is assumed to be a lattice $(\Sigma, \preceq)$, whose elements are ranged over by ρ, σ, τ. Based on an assignment γ of security levels to subject and objects, one defines a *security policy* as a ternary boolean predicate $\mathcal{P}$ on *subject levels*, *object levels*, and *access modes* $\mathcal{A}, \mathcal{B} \in \{\mathsf{w}, \mathsf{r}, \mathsf{rw}, -\}$ (the mode "$-$", denoting "no access", is introduced for convenience, to make the notation uniform). Specifically, an access $\mathcal{A}$ to an object o by a subject s is legal under $\mathcal{P}$ if and only if $\mathcal{P}(\gamma(s), \gamma(o), \mathcal{A})$ holds true. Military security (no read-up, no write-down) and commercial security (no read-up, no write-up) can be enforced by the following security policies:

$$
\begin{array}{lcl@{\qquad}lcl}
\mathcal{P}_{Mil}(\rho,\sigma,\mathsf{r}) & \triangleq & \sigma \preceq \rho & \mathcal{P}_{Com}(\rho,\sigma,\mathcal{A}) & \triangleq & \sigma \preceq \rho \quad \mathit{for} \quad \mathcal{A} \in \{\mathsf{r},\mathsf{w},\mathsf{rw}\} \\
\mathcal{P}_{Mil}(\rho,\sigma,\mathsf{w}) & \triangleq & \rho \preceq \sigma & \mathcal{P}_{Com}(\rho,\sigma,-) & \triangleq & \mathit{true} \\
\mathcal{P}_{Mil}(\rho,\sigma,\mathsf{rw}) & \triangleq & \sigma = \rho & & & \\
\mathcal{P}_{Mil}(\rho,\sigma,-) & \triangleq & \mathit{true} & & &
\end{array}
$$

9.1 Subjects, objects and security levels for Boxed Ambients

We study this form of access control for the asynchronous version of the calculus discussed in the previous section. Processes take the role of subjects, while ambients take the role of objects, and we rely on the notion of access we have assumed throughout, namely: $(x)^n P$ and $\langle M\rangle^n P$ represent processes (subjects) attempting to access the child n in read and write mode, respectively, whereas $\langle M\rangle^{\uparrow} P$ and $(x)^{\uparrow} P$ represent processes requesting an access to their parent ambient, again in read and write mode. We make these notions formal, and define the import of a security policy in the calculus by introducing a tagged version of the asynchronous reduction relation in which any unauthorized access to an ambient results into a distinguished error-reduction.

The relation of tagged reduction is denoted $\longrightarrow_{(\sigma,\gamma)}$ and defined in terms of a security environment γ that associates names (and variables) to security levels, and a security level σ that identifies the clearance of the processes involved in the reduction. Security environments are formed just as typing environments, according to the following production:

$$\gamma ::= \varnothing \;\mid\; \gamma, x:\sigma$$

where in $\gamma, x:\sigma$ it is understood that $x \notin Dom(\gamma)$. Based on that, we take any reduction to a distinguished process term err as the formal counterpart of an access violation.

The reductions, summarized in Table VI, should be understood easily. In particular, note that in the reductions for non-local input/output the clearance of a process enclosed in an ambient is determined by the security level associated with the ambient's name. This is consistent with the format of the reduction (AMB). To illustrate, consider

$$P \;\equiv\; h[\,\ell[\,\mathsf{out}\,h.\mathsf{in}\,h.(x)^{\uparrow}Q \mid \langle M\rangle\,]\,] \mid (y)^{\ell}R$$

where $\gamma(h) = \top$ and $\gamma(\ell) = \bot$, and let $\sigma \succeq \bot$ be any security level. Then we have

$$\begin{array}{rll}
P & \longrightarrow_{(\sigma,\gamma)} h[\,] \mid \ell[\mathsf{in}\,h.(x)^{\uparrow}Q \mid \langle M\rangle] \mid (y)^{\ell}R & \text{by (EXIT)} \\
 & \longrightarrow_{(\sigma,\gamma)} h[\,] \mid \ell[\mathsf{in}\,h.(x)^{\uparrow}Q] \mid R\{x := M\} & \text{by (INPUT } n) \\
 & \longrightarrow_{(\sigma,\gamma)} h[\,\ell[\,(x)^{\uparrow}Q]\,] \mid R\{x := M\} & \text{by (ENTER)} \\
 & \longrightarrow_{(\sigma,\gamma)} \mathsf{err} \quad \text{by (ERR INPUT} \uparrow\text{), (ERR AMB), (ERR PAR)} &
\end{array}$$

Here, the error reduction arises from the low-level process inside ℓ attempting to read from h, a top-level ambient. As we discuss next, this illegal attempt is detected statically by the type system. It is worth remarking that there is *no* dynamic access control intended in the tagged reduction: access control will be provided statically, by typing, and the tagged semantics is only defined to give a formal statement of the soundness result for the type system.

Finally, note that reductions to err only result from attempts to read or write on non-local resources. As such, we disregard errors resulting from type mismatches in any of the local or non-local exchanges of values (in fact, for well-typed processes the absence of such errors is guaranteed by subject reduction).

9.2 Access Control Types and Typing rules

The new classes of types extend the types of Section 7 with new tags specifying the security clearance of capabilities and ambient names, and additional information of the on the read

Table VI Asynchronous Tagged Reduction

Top-level reductions

(INPUT $\star$)	$(\tilde{x}:\tilde{W})P \mid \langle \tilde{M} \rangle$	$\longrightarrow_{(\sigma,\gamma)}$	$P\{\tilde{x} := \tilde{M}\}$	
(INPUT n)	$(\tilde{x}:\tilde{W})^{n}P \mid n[\langle \tilde{M} \rangle \mid Q]$	$\longrightarrow_{(\sigma,\gamma)}$	$P\{\tilde{x} := \tilde{M}\} \mid n[Q]$	*if* $\mathcal{P}(\sigma,\gamma(n),\mathsf{r})$
(ERR INPUT n)	$(\tilde{x}:\tilde{W})^{n}P \mid n[Q]$	$\longrightarrow_{(\sigma,\gamma)}$	err	*if* $\neg\mathcal{P}(\sigma,\gamma(n),\mathsf{r})$
(INPUT $\uparrow$)	$n[(\tilde{x}:\tilde{W})^{\uparrow}P \mid Q] \mid \langle \tilde{M} \rangle$	$\longrightarrow_{(\sigma,\gamma)}$	$n[P\{\tilde{x} := \tilde{M}\} \mid Q]$	*if* $\mathcal{P}(\gamma(n),\sigma,\mathsf{r})$
(ERR INPUT $\uparrow$)	$n[(\tilde{x}:\tilde{W})^{\uparrow}P \mid Q]$	$\longrightarrow_{(\sigma,\gamma)}$	err	*if* $\neg\mathcal{P}(\gamma(n),\sigma,\mathsf{r})$
(OUTPUT n)	$\langle \tilde{M} \rangle^{n} \mid n[P]$	$\longrightarrow_{(\sigma,\gamma)}$	$n[\langle \tilde{M} \rangle \mid P]$	*if* $\mathcal{P}(\sigma,\gamma(n),\mathsf{w})$
(ERR OUTPUT n)	$\langle \tilde{M} \rangle^{n} \mid n[P]$	$\longrightarrow_{(\sigma,\gamma)}$	err	*if* $\neg\mathcal{P}(\sigma,\gamma(n),\mathsf{w})$
(OUTPUT $\uparrow$)	$n[\langle \tilde{M} \rangle^{\uparrow}Q \mid R]$	$\longrightarrow_{(\sigma,\gamma)}$	$\langle \tilde{M} \rangle \mid n[Q \mid R]$	*if* $\mathcal{P}(\gamma(n),\sigma,\mathsf{w})$
(ERR OUTPUT $\uparrow$)	$n[\langle \tilde{M} \rangle^{\uparrow}Q \mid R]$	$\longrightarrow_{(\sigma,\gamma)}$	err	*if* $\neg\mathcal{P}(\gamma(n),\sigma,\mathsf{w})$
(ENTER)	$a[\mathsf{in}\ b.P \mid Q] \mid b[R]$	$\longrightarrow_{(\sigma,\gamma)}$	$b[a[P \mid Q] \mid R]$	
(EXIT)	$a[b[\mathsf{out}\ a.P \mid Q] \mid R]$	$\longrightarrow_{(\sigma,\gamma)}$	$b[P \mid Q] \mid a[R]$	

Structural Reductions: the symmetric reductions for (Par) and (Err Par) are omitted. The rules (New) and (Err New) assume $A = \rho\mathsf{Amb}^{?}[\_,\_]$

(STRUCT)	$P' \equiv P \quad P \longrightarrow_{(\sigma,\gamma)} Q \quad Q \equiv Q' \;\Rightarrow\; P' \longrightarrow_{(\sigma,\gamma)} Q'$
(ERR STRUCT)	$Q \equiv P \quad P \longrightarrow_{(\sigma,\gamma)} \mathsf{err} \;\Rightarrow\; Q \longrightarrow_{(\sigma,\gamma)} \mathsf{err}$
(NEW)	$P \longrightarrow_{(\sigma,(\gamma,n:\rho))} Q \;\Rightarrow\; (\nu n{:}A)P \longrightarrow_{(\sigma,\gamma)} (\nu n{:}A)Q$
(ERR NEW)	$P \longrightarrow_{(\sigma,(\gamma,n:\rho))} \mathsf{err} \;\Rightarrow\; (\nu n{:}A)P \longrightarrow_{(\sigma,\gamma)} \mathsf{err}$
(PAR)	$P \longrightarrow_{(\sigma,\gamma)} Q \;\Rightarrow\; P \mid R \longrightarrow_{(\sigma,\gamma)} Q \mid R$
(ERR PAR)	$P \longrightarrow_{(\sigma,\gamma)} \mathsf{err} \;\Rightarrow\; P \mid R \longrightarrow_{(\sigma,\gamma)} \mathsf{err}$
(AMB)	$P \longrightarrow_{(\rho,\gamma)} Q \quad \rho = \gamma(a) \;\Rightarrow\; a[P] \longrightarrow_{(\sigma,\gamma)} a[Q]$
(ERR AMB)	$P \longrightarrow_{(\gamma(a),\gamma)} \mathsf{err} \;\Rightarrow\; a[P] \longrightarrow_{(\sigma,\gamma)} \mathsf{err}$

and write access requests on such names.

$$\begin{array}{lll}
\textit{Ambient Types} & A & ::= \ \sigma\mathsf{Amb}[E,F,\mathcal{A}] \mid \sigma\mathsf{Amb}^{\circ}[E,F,\mathcal{A}] \\
\textit{Expression Types} & W & ::= \ A \ \mid \ \sigma\mathsf{Cap}[E,\mathcal{A}] \\
\textit{Exchange Types} & E,F & ::= \ \mathsf{shh} \quad \mid \quad W_1 \times \cdots \times W_n \\
\textit{Process Types} & T & ::= \ \mathsf{Pro}[E,F,\mathcal{A}] \ \mid \ \mathsf{Pro}[E,^{\mu}F,\mathcal{A}]
\end{array}$$

The new types are interpreted similarly to the types we introduced in the previous sections. In particular, the exchange components E and F (with their modes μ) have the same interpretation as in Section 7. The meaning of the new components is as follows. In $\sigma\mathsf{Amb}[E,F,\mathcal{A}]$, σ is the clearance of ambients with this type, and $\mathcal{A}$ is (a sound estimate of) the access mode of the upward exchanges of their enclosing processes. Similarly, $\sigma\mathsf{Cap}[F,\mathcal{A}]$ is the type of a capability that can be exercised within an ambient with clearance σ, upward exchanges of type F and access mode $\mathcal{A}$. Finally, $\mathsf{Pro}[E,^{\mu}F,\mathcal{A}]$ is the type of a process with local and upward exchanges of type, respectively E and F, and access mode $\mathcal{A}$. Note that process types are associated with two modes: the access mode $\mathcal{A}$ defines the mode in which the process accesses the channel located in its parent ambient; the exchange mode μ defines whether the process is silent, moving or both. We do not explicitly associate security levels with process types: instead, we type check processes at a given security level, by introducing judgments of the form $\Gamma \vdash_{\sigma} P : T$, where T is process type, and σ a security level. The typing rules are collected in Appendix A: most of them are the direct generalization of the corresponding rules in Section 7, and so are most of the rules for processes, with the exceptions discussed below. We assume the following partial ordering on access modes: $- \leqslant \{\mathsf{r},\mathsf{w}\} \leqslant \mathsf{rw}$.

9.2.1 *Typing of Capabilities.* In addition to the usual type safety constraints, rules (IN) and (OUT) introduce the constraints relative to the security policy under consideration.

(IN)
$$\frac{\Gamma \vdash M : \sigma\mathsf{Amb}^{?}[E,F,\mathcal{B}] \quad \mathcal{P}(\rho,\sigma,\mathcal{A}) \quad G \leqslant E}{\Gamma \vdash \mathsf{in}\, M : \rho\mathsf{Cap}[G,\mathcal{A}]}$$

(OUT)
$$\frac{\Gamma \vdash M : \sigma\mathsf{Amb}[E,F,\mathcal{B}] \quad G \leqslant F,\ \mathcal{A} \leqslant \mathcal{B}}{\Gamma \vdash \mathsf{out}\, M : \rho\mathsf{Cap}[G,\mathcal{A}]}$$

Specifically, an "in" reduction is well-typed only if the security levels of the two ambients involved in the move are compatible. Dually, an out move type checks only if the type of the upward exchanges of the exiting ambient are already encompassed by the upward component of the type of the exited ambient. The rule (OUT) specializes in the natural way to the case in which M is a pilot ambient (cf. Appendix A).

9.2.2 *Typing of Prefixes.* The typing rules for prefixes have the same rationale as those of the moded typing system. For example,

(PREFIX ∘)
$$\frac{\Gamma \vdash^{\circ} M : \rho\mathsf{Cap}[G,\mathcal{B}] \quad \Gamma \vdash_{\sigma} P : \mathsf{Pro}[E,^{\circ}F,\mathcal{A}]}{\Gamma \vdash_{\sigma} M.P : \mathsf{Pro}[E,^{\circ}F,\mathcal{A}]}$$

states that we can safely disregard the access mode $\mathcal{B}$ whenever the process and, therefore, the pilot ambient containing it are in a silent phase.

9.2.3 *Typing of Ambients.* The typing rules for ambients define the clearance level at which the enclosed processes should be type-checked: if P is enclosed into a σ-level ambient, then P is type-checked at clearance σ. In addition, the rules predicate well-typing to the security policy under consideration. A representative of these rules is the rule (AMB):

(AMB)
$$\frac{\Gamma \vdash a : \sigma\mathsf{Amb}[E,F,\mathcal{A}] \quad \Gamma \vdash_{\sigma} P : \mathsf{Pro}[E,F,\mathcal{A}] \quad \mathcal{P}(\sigma,\rho,\mathcal{A})}{\Gamma \vdash_{\rho} a[P] : \mathsf{Pro}[F,^{\bullet}H,\mathcal{B}]}$$

The constraint imposed by the policy $\mathcal{P}$ can safely be lifted for ambients whose enclosing processes are moving (or silent), as in that case there is no upward access to be checked (cf. the (AMB ∘) rule in Appendix A).

9.2.4 *Typing of input/output.* The rules for local exchanges are straightforward: they enforce no access control, as processes are always granted access to their local resources. The rules for downward input/output relate the types of the input-output processes and their continuations, as in the type system of Section 7. In addition, they enforce the constraint that processes at clearance σ read only from (rule INPUT M) and write only to (rule OUTPUT M) ambients of clearance ρ compatible with σ according to the given security policy. The rule for upward exchange do not impose any access control and just check that the access modes are correct: this is sound, as the upward accesses are already regulated by the rules for ambients, and by the rules governing mobility.

9.2.5 *Subtyping and Subsumption.* Subtyping over exchange and process types extends uniformly to the new set of types. As in the type systems of the previous sections, subtyping is only reflexive on capability and ambient types. Process subtyping, in turn, is the direct extension of the subtyping relation of Section 7, defined as follows:

$$\frac{\mathsf{Pro}[E,^{\mu_1}F] \leqslant_7 \mathsf{Pro}[E',^{\mu_2}F]}{\mathsf{Pro}[E,^{\mu_1}F,\mathcal{A}] \leqslant \mathsf{Pro}[E',^{\mu_2}F,\mathcal{A}]}$$

where μ_i are (possibly empty) modes, and $\leqslant_7$ is the subtyping relation for processes defined in Section 7. Richer subtyping relations, such as one allowing a relation $\mathsf{Pro}[E,F,\mathsf{r}] \leqslant \mathsf{Pro}[E,F,\mathsf{rw}]$, would be desirable, but turn out to be unsound. To see the problem with this form of subtyping, consider a system with military security and two security levels, $\top$ and $\bot$ with $\bot \prec \top$. Take then Γ to be a type environment such that $\Gamma(\ell) = \bot\mathsf{Amb}[W,\mathsf{shh},\mathsf{r}]$ and $\Gamma(h) = \top\mathsf{Amb}[\mathsf{shh},W,\mathsf{rw}]$. Under these assumptions, the judgment $\Gamma \vdash_\top \mathsf{in}\ \ell.\mathbf{0} : \mathsf{Pro}[\mathsf{shh},W,\mathsf{r}]$ is derivable by the type system of this section for any W. If, by subtyping, we upgrade the previous typing judgment to $\Gamma \vdash_\top \mathsf{in}\ \ell.\mathbf{0} : \mathsf{Pro}[\mathsf{shh},W,\mathsf{rw}]$, and take M of type W, the following judgment is derivable: $\Gamma \vdash_\top \mathsf{in}\ \ell.\mathbf{0} \mid \langle M\rangle^\uparrow : \mathsf{Pro}[\mathsf{shh},W,\mathsf{rw}]$. From this judgment, and from the assumption $\Gamma(h) = \top\mathsf{Amb}[\mathsf{shh},W,\mathsf{rw}]$, we then have $\Gamma \vdash_\top h[\mathsf{in}\ \ell.\mathbf{0} \mid \langle M\rangle^\uparrow\,] : \mathsf{Pro}[W,F,\mathcal{A}]$, for any type F and mode $\mathcal{A}$. This typing is unsound, however, because h can move into ℓ and make a write access to the low-level ambient ℓ, thus violating the military security policy we had assumed.

9.3 Soundness of the Type System

The main purpose of the type system of this section is to statically detect access violations, with respect to the underlying security policy. As we state below (and prove in Appendix B), the type system does provide these guarantees under the additional hypothesis that the security policy is *stable*, in the sense of the following definition.

DEFINITION STABLE SECURITY POLICIES. *We say that a security policy $\mathcal{P}$ is stable if and only if it satisfies the following conditions*

(*1*) *if* $\mathcal{P}(\sigma,\rho,\mathcal{A})$ *and* $\mathcal{P}(\rho,\tau,\mathcal{A})$ *then* $\mathcal{P}(\sigma,\tau,\mathcal{A})$.

(*2*) *if* $\mathcal{P}(\sigma,\rho,\mathcal{A})$ *and* $\mathcal{C} \leqslant \mathcal{A}$ *then* $\mathcal{P}(\sigma,\rho,\mathcal{C})$. $\square$

Military and Commercial security, as defined in this section, are both examples of stable policies.

Given a type environment Γ, and a security assignment γ we say that γ is Γ-*consistent* if and only if for all $x \in dom(\Gamma)$, $\Gamma(x) = \sigma\mathsf{Amb}^?[\ldots]$ implies $\gamma(x) = \sigma$. We use this definition to state the soundness of our type system:

THEOREM TYPE SOUNDNESS. *Assume that* $\Gamma \vdash_\sigma P : T$. *Then for every* Γ*-consistent security assignment* γ*, and process* Q *such that* $P \longrightarrow^*_{(\sigma,\gamma)} Q$ *we have* $Q \not\longrightarrow_{(\sigma,\gamma)} \mathsf{err}$.

To exemplify the effects of typing, consider the following example from Section 8.

$$h[\ \ell[\,(x)^{\uparrow}P\,] \mid \ell'[\,\langle M\rangle^{\uparrow}Q\,]\,].$$

If we assume that the clearance of ambient h is strictly higher than the clearance of ℓ and ℓ', then a type system based on a "no read-up" policy should reject the above process as ill-typed, because not secure. This is indeed the case for our type system: to see that, note that the process $\ell[\,(x)^{\uparrow}P\,]$ is type checked at the clearance of the enclosing ambient h, and the side condition to the (AMB Δ) rules fails to be satisfied under a "no read-up" policy.

A similar reasoning shows that the unsafe process

$$h[\ell[\,\mathsf{out}\ h.\mathsf{in}\ h.(x{:}W)^{\uparrow} \mid \langle N\rangle\,] \mid \langle M\rangle\,] \mid (y)^{\ell}$$

where ℓ and h have clearance $\bot$ and $\top$ respectively, is ill-typed, as ℓ inside h performs a read up-operation. In particular, in order for P to type check, the subprocess $\mathsf{in}\ h.(x{:}W)^{\uparrow}$ enclosed in ℓ can only be typed at level $\bot$ with the process type $\mathsf{Pro}[E,{}^{\mu}W,\mathcal{A}]$, for $\mathsf{r} \leqslant \mathcal{A}$. However, the judgment $\Gamma \vdash_{\bot} \mathsf{in}\ h.(x{:}W)^{\uparrow} : \mathsf{Pro}[E,{}^{\mu}W,\mathcal{A}]$ must come from $\Gamma \vdash \mathsf{in}\ h : \bot\mathsf{Cap}[W,\mathcal{A}]$, which is not derivable since the clearance $\bot$ of the ambient h is greater than the clearance of ℓ, contradicting the hypothesis of rule (IN).

9.4 Discussion

The notion of access control encompassed by the type system, and the corresponding notion of type soundness we have addressed can be further strengthened. To motivate, consider the following program:

$$h[\,\langle secret\rangle \mid \ell[\,(\nu t)t[\,\mathsf{out}\ h.(x)^{\uparrow}\mathsf{in}\ \ell.\langle x\rangle\,]\,] \mid (x)^{t}P\,]$$

operating under commercial security (no-read-up/no-write-up). If we assume that h and ℓ have clearance, respective, $\top$ and $\bot$, commercial security should prevent ℓ to read the *secret* from h. In fact, our type system does provide this guarantee. For, if t is given clearance $\top$, then the type system rejects $(x)^{t}P$ as ill-typed (because $(x)^{t}$ is a read-up). If instead, $t : \bot$, the type failure arises at $(x)^{\uparrow}\mathsf{in}\ \ell\ldots$, which is also classified as read-up in this case. However, one can play the following trick:

$$h[\,\langle secret\rangle \mid \ell[\,(\nu t : \top)t[\,\mathsf{out}\ h.(x)^{\uparrow}\mathsf{in}\ \ell.\langle x\rangle^{\uparrow}\,]\,] \mid (x)P\,]$$

The ambient ℓ creates a Trojan horse t that is entitled to exchange values with h: now t reads the *secret*, and then, once back into ℓ, it delivers it upward, to P. This program type-checks in our system, and indeed, there is no reduction to err involved in the computation, as none of the read and write operation violate any access control constraint.

On the other hand, it may reasonably be argued that one should prevent such situations. In fact, the type system may easily be extended to account for the cases of interest. If we look at the example, we notice that the problem arises as a result of the *secret*, a high value, being communicated to the ambient ℓ, which is low. In the current system this goes unnoticed, as the typing rules enforce no constraint relationship between the security levels of an ambient and those of the values the ambient may exchange. Such constraints are easily incorporated in the type system by means of new typing formation rules. Given any exchange type E, let $\lambda(E)$ denote the security level associated with E, defined as follows:

$$\lambda(\sigma\mathsf{Amb}[E,F]) = \lambda(\sigma\mathsf{Cap}[E]) = \sigma\ ,\ \lambda(W_1 \times \cdots \times W_n) = \lambda(W_1) \sqcup \cdots \sqcup \lambda(W_n)\ ,\ \lambda(\mathsf{shh}) = \bot$$

Based on that, we can define type formation rules enforcing the expected constraints, namely that ambients (and processes) of clearance σ may only exchange values of clearance at most σ.

$$\text{(ENV } \oslash\text{)} \quad \frac{}{\varnothing \vdash \diamond} \qquad \text{(ENV } n\text{)} \quad \frac{\Gamma \vdash W \quad n \notin \mathit{Dom}(\Gamma)}{\Gamma, n : W \vdash \diamond} \qquad \text{(TYPE shh)} \quad \frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathsf{shh}}$$

$$\text{(TYPE CAP)} \quad \frac{\Gamma \vdash E}{\Gamma \vdash \sigma\mathsf{Cap}[E]} \qquad \text{(TYPE AMB)} \quad \frac{\Gamma \vdash E_i \quad \lambda(E_i) \preceq \sigma \quad i = 1,2}{\Gamma \vdash \sigma\mathsf{Amb}[E_1, E_2]} \qquad \text{(TYPE PROC)} \quad \frac{\Gamma \vdash E_i \quad \lambda(E_i) \preceq \sigma \quad i = 1,2}{\Gamma \vdash_\sigma \mathsf{Pro}[E_1, E_2]}$$

These rules can be incorporated in the type system, so as to ensure that all types used in a derivation are well-formed. With the extended system the last example would be rejected, because the high value *secret* may no longer be exchanged with a low-level ambient like ℓ. There is still the possibility for ℓ to exploit t to obtaining access to data stored in h, but only the only data that can eventually flow to ℓ must have low level, which is acceptable.

9.5 Examples

We demonstrate the import of the type system in enforcing effective access control policies on several examples.

9.5.1 *Wrappers.* As a solution for the problem of resource protection and access control in wide-area networks, Sewell and Vitek [Sewell and Vitek 2000] propose an extension of the π-calculus, known as the *Box* π-calculus. Within this calculus, they develop a programming technique, based on *wrappers*, whereby untrusted code can be secured into an isolated *box*, and its interactions with the enclosing environment filtered by a process, the *wrapper*, that only forwards legitimate messages between the boxed program and its enclosing environment via secured channels. The paradigmatic example of that work can be rephrased in our syntax as follows:

$$(\nu a, b)\Big(a[P] \mid !(x)^a\langle x\rangle^b \mid b[Q]\Big).$$

P and Q are arbitrary processes that are encapsulated in ambients (“named boxes” in the terminology of [Sewell and Vitek 2000]) with private names a and b, and placed in parallel with a process that forwards messages from a to b. Notice that ambient boundaries prevent any direct interaction between P and Q, and the name restrictions on a and b ensure that the only possible exchanges with the environment are filtered by the process $!(x)^a\langle x\rangle^b$. Thus, as in Boxed-π, we can rely on wrappers to provide interesting security guarantees: specifically, the above configuration prevents (*i*) Q from leaking secrets to P and (*ii*) P and Q from corrupting the environment.

With the type system of Section 9, we can provide further guarantees. If we define a to be a high-level ambient and b a low-level ambient, then the type system built over military security will detect any unwanted access from Q to P regardless of context that encloses $a[P]$ and $b[Q]$. In addition, military security may be employed in the type system also to detect any attempt by P and Q to access the environment: for that purpose, we only need to type-check the configuration at a clearance incomparable with the clearance of a and b.

9.5.2 *Firewalls.* We now look at the protocol for firewall crossing defined in [Cardelli and Gordon 1999b] and refined in [Levi and Sangiorgi 2000]. The protocol can be expressed in our calculus as follows:

$$\begin{aligned} \mathit{Firewall} &= (\nu f)f[k[\mathsf{out}\ f.\langle \mathsf{in}\ f\rangle^a] \mid \ldots] \\ \mathit{Agent} &= a[\mathsf{in}\ k.(x)\mathsf{out}\ k.x.Q] \end{aligned}$$

The idea is to let the *Agent* a cross the *Firewall* f by means of a shared key k. In [Cardelli and Gordon 1999b], the key k is used as the name of a pilot ambient that drives the agent into the protocol and is then dissolved. Our coding follows the same idea, but implements it differently, relying on communication: a enters k from which it receives the capability $\mathsf{in}\ f$ to be exercised after $\mathsf{out}\ k$ to drive a into the firewall. Interestingly, the process $a[\mathsf{in}\ k.(x)\mathsf{out}\ k.x.Q]$, where a capability is first read (locally) and then exercised at the same nesting level is well-typed in our system (this is not true of the type system of [Cardelli and Gordon 1999b]).

Having authenticated an incoming agent, the firewall may then provide other security guarantees. For example, we may want to ensure that processes inside the firewall can access the resources of the agent that crossed the firewall, but not the converse. This guarantee can be provided with commercial security, by the type assignments $f : \phi\mathsf{Amb}[E,F,\mathcal{A}]$ and $k : \kappa\mathsf{Amb}[\mathsf{shh},\mathsf{shh}]$, where $\kappa \prec \phi$, E and F are appropriate types, and $\mathcal{A}$ is an appropriate access right (note that this constraint are defined just for the firewall, independently from the interacting agent).

To illustrate the effect of these type assignments, consider a generic agent $a[P]$ entering the firewall, and assume that $a : \alpha\mathsf{Amb}[G,H,\mathcal{B}]$. To cross the firewall, a must accept write requests from k: with commercial security, this is possible only if $\alpha \preceq \kappa$ and this, by transitivity, implies that $\alpha \prec \phi$. Now observe that commercial security prevents low-level ambients (such as a) contained in high-level ambients (such as f) from attempting upward exchanges. Then, $\alpha \prec \phi$ implies $H = \mathsf{shh}$. In summary, the type assignment enforces on a a security level strictly smaller than the level of f, and this implies that any agent entering the firewall f cannot directly access to local resources of f, as desired.

The protocol we just discussed depends on the assumption that the firewall knows the name of the incoming agent. To overcome the problem, we may either assume that a itself is also a password which is part of the protocol, or devise new protocol that relies on two passwords, k and h, and structure the agent and the firewall as follows:

$$\begin{aligned} \mathit{Firewall} &= (\nu f)f[k[\mathsf{out}\ f.\langle \mathsf{in}\ f\rangle^h] \mid \ldots] \\ \mathit{Agent} &= h[\mathsf{in}\ k.(x)\mathsf{out}\ k.\langle x\rangle^a \mid a[(x).\mathsf{out}\ h.x.Q]\,] \end{aligned}$$

10. DISTRIBUTED LANGUAGE SECURITY

We conclude with a more extended example that illustrates the access control typing system on a simple, but non-trivial distributed language. The language is defined in [Cardelli et al. 2000], with the following syntax.

The computational model is that of various distributed variant of the π calculus in the literature, such as those described by Amadio and Prasad [Amadio and Prasad 1994] and Hennessy and Riely's Dπ calculus [Riely and Hennessy 1998]. A network consists of named nodes that contain named channels and anonymous threads (node and channel names can be restricted). Channels are represented as persistent resources, as suggested in Section 4. Threads are the active components of a network. They can move across nodes of the

Table VII A simple distributed language

Types	$W ::= \mathit{Node}$	type of nodes
	$\mid \mathit{Ch}(W)$	type of channels
Network	$\mathit{Net} ::= \texttt{node}\ n[\mathit{Cro}]$	node
	$\mid (\nu n{:}W)\mathit{Net}$	restriction
	$\mid \mathit{Net} \mid \mathit{Net}$	network composition
Crowd	$\mathit{Cro} ::= \texttt{channel}\ c$	channel
	$\mid \texttt{thread}\ [\mathit{Th}]$	thread
	$\mid \mathit{Cro} \mid \mathit{Cro}$	crowd composition
	$\mid (\nu c{:}W)\mathit{Cro}$	restriction
Threads	$\mathit{Th} ::= \texttt{go}\ n.\mathit{Th}$	migration
	$\mid \overline{c}(x)$	output to a channel
	$\mid c(x).\mathit{Th}$	input from a channel
	$\mid \texttt{fork}\ (\mathit{Cro}).\mathit{Th}$	fork a new crowd
	$\mid \texttt{spawn}\ n[\mathit{Cro}].\mathit{Th}$	spawn a new node

network, communicate over local channels (i.e. residing on the same node), fork into a new set of channels and threads or spawn a new node. The type system of the language presents no surprise: threads are well-typed if (i) their access to channels are well-typed and (ii) they fork and spawn well-typed crowds and networks. The latter are well typed if they are formed by well-typed subcomponents. We omit the typing rules, and refer the interested reader to [Cardelli et al. 2000] for further details.

We illustrate the semantics (and the typing) of the language with the following program:

$$\begin{array}{l}\texttt{node}\ n[\texttt{channel}\ \mathit{rcv}\mid\\ \qquad (\nu c{:}W)\texttt{thread}[\mathit{rcv}(x).\texttt{fork}(\texttt{channel}\ c\mid\texttt{thread}[c(y).P]).\texttt{go}\ x.\overline{\mathit{ack}}(c)]]\end{array}$$

which is well-typed under the hypotheses $n{:}\mathit{Node}$, $\mathit{rcv}{:}\mathit{Ch}(\mathit{Node})$, and $\mathit{ack}{:}\mathit{Ch}(\mathit{Ch}(W))$. The program simulates the behaviour of a dæmon that listens on some public port *rcv* and once contacted, it forks to establish the connection on some other private port. The dæmon is located at node n, containing the channel *rcv* and a single thread. The thread waits for a request on *rcv*; subsequently it spawns a private channel c to be shared with the node x communicated on *rcv* and a new thread that listens on this channel; finally it communicates the private name c to the node x. The dæmon will typically interact with remote clients of the following structure:

$$\texttt{node}\ m[\ \texttt{channel}\ \mathit{ack}\ \mid\ \texttt{thread}[\texttt{fork}(\texttt{thread}[\texttt{go}\ n.\overline{\mathit{rcv}}(m)]).\mathit{ack}(y).\ \ldots]\ ]$$

The client is located at node m, which allocates the channel *ack*, and spawns a thread that first forks into a new thread that moves to n to communicate over the channel *rcv* the name of its origin node m, and then waits to receive on channel *ack* the private name on which to establish the communication with n.

We give a typed encoding for this language by relying on the typing system of Section 9. The encoding is defined in Figure VIII. We initially assume that all types (from the source language) have the same security level. Later, we will study new encodings for enhanced versions of the source language in which we associate different levels with channels, nodes, and threads. To ease the notation, we write $\sigma\mathsf{Amb}[E]$ for $\sigma\mathsf{Amb}[E,\mathsf{shh},\mathcal{A}]$, omitting the security level σ when it is not relevant to the discussion. Also, we use *Synch* to denote the empty tuple type.

Table VIII Encoding of the distributed language

Types :	
$\langle\!\langle\, \mathit{Node} \,\rangle\!\rangle$	$= \mathsf{Amb}[\mathsf{shh}]$
$\langle\!\langle\, \mathit{Ch}(W) \,\rangle\!\rangle$	$= \mathsf{Amb}[\,\langle\!\langle\, W \,\rangle\!\rangle\,]$
Net :	
$\langle\!\langle\, (\nu n{:}\mathit{Node})\mathit{Net} \,\rangle\!\rangle_\Gamma$	$= (\nu n{:}\,\langle\!\langle\, \mathit{Node} \,\rangle\!\rangle\,)\,\langle\!\langle\, \mathit{Net} \,\rangle\!\rangle_\Gamma$
$\langle\!\langle\, (\nu c{:}\mathit{Ch}(W))\mathit{Net} \,\rangle\!\rangle_\Gamma$	$= (\nu c{:}\,\langle\!\langle\, \mathit{Ch}(W) \,\rangle\!\rangle\,)\,\langle\!\langle\, \mathit{Net} \,\rangle\!\rangle_{\Gamma,c:W}$
$\langle\!\langle\, \mathit{Net}_1 \mid \mathit{Net}_2 \,\rangle\!\rangle_\Gamma$	$= \langle\!\langle\, \mathit{Net}_1 \,\rangle\!\rangle_\Gamma \mid \langle\!\langle\, \mathit{Net}_2 \,\rangle\!\rangle_\Gamma$
$\langle\!\langle\, \texttt{node}\ n[\mathit{Cro}] \,\rangle\!\rangle_\Gamma$	$= n[\,\langle\!\langle\, \mathit{Cro} \,\rangle\!\rangle^n_\Gamma\,]$
Crowd :	
$\langle\!\langle\, (\nu m{:}\mathit{Node})\mathit{Cro} \,\rangle\!\rangle^n_\Gamma$	$= (\nu m{:}\,\langle\!\langle\, \mathit{Node} \,\rangle\!\rangle\,)\,\langle\!\langle\, \mathit{Cro} \,\rangle\!\rangle^n_\Gamma$
$\langle\!\langle\, (\nu c{:}\mathit{Ch}(W))\mathit{Cro} \,\rangle\!\rangle^n_\Gamma$	$= (\nu c{:}\,\langle\!\langle\, \mathit{Ch}(W) \,\rangle\!\rangle\,)\,\langle\!\langle\, \mathit{Cro} \,\rangle\!\rangle^n_{\Gamma,c:W}$
$\langle\!\langle\, \mathit{Cro}_1 \mid \mathit{Cro}_2 \,\rangle\!\rangle^n_\Gamma$	$= \langle\!\langle\, \mathit{Cro}_1 \,\rangle\!\rangle^n_\Gamma \mid \langle\!\langle\, \mathit{Cro}_2 \,\rangle\!\rangle^n_\Gamma$
$\langle\!\langle\, \texttt{thread}\ [\mathit{Th}] \,\rangle\!\rangle^n_\Gamma$	$= (\nu t{:}\ \mathsf{Amb}[\mathsf{shh}])t[\,\langle\!\langle\, \mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma\,]$
$\langle\!\langle\, \texttt{channel}\ c \,\rangle\!\rangle^n_\Gamma$	$= c[\,!(x{:}\,\langle\!\langle\, \Gamma(c) \,\rangle\!\rangle\,)\langle x\rangle\,]$
Threads :	
$\langle\!\langle\, \bar{c}(x) \,\rangle\!\rangle^{n,t}_\Gamma$	$= (\nu w{:}\ \mathsf{Amb}^\circ[\mathsf{shh},\langle\!\langle\, \Gamma(c) \,\rangle\!\rangle,\mathsf{w}])w[\mathsf{out}\ t.\mathsf{in}\ c.\langle x\rangle^\uparrow]$
$\langle\!\langle\, c(x).\mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$	$= (\nu r{:}\ \mathsf{Amb}^\circ[\,\langle\!\langle\, \Gamma(c) \,\rangle\!\rangle,\langle\!\langle\, \Gamma(c) \,\rangle\!\rangle,\mathsf{r}])\ (x{:}\,\langle\!\langle\, \Gamma(c) \,\rangle\!\rangle\,)^r\,\langle\!\langle\, \mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$
	$\mid r[\mathsf{out}\ t.\mathsf{in}\ c.(x{:}\,\langle\!\langle\, \Gamma(c) \,\rangle\!\rangle\,)^\uparrow.\mathsf{out}\ c.\mathsf{in}\ t.\langle x\rangle\,]$
$\langle\!\langle\, \texttt{go}\ m.\mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$	$= \mathsf{out}\ n.\mathsf{in}\ m.\,\langle\!\langle\, \mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$
$\langle\!\langle\, \texttt{fork}\ (\texttt{channel}(c)).\mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$	$= (\nu s{:}\ \mathsf{Amb}[\mathit{Synch}])\ ()^s\,\langle\!\langle\, \mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$
	$\mid\ c[\mathsf{out}\ t.(!((x{:}\,\langle\!\langle\, \Gamma(c) \,\rangle\!\rangle\,)\langle x\rangle) \mid s[\mathsf{out}\ c.\mathsf{in}\ t.\langle\rangle\,]\,)\,]$
$\langle\!\langle\, \texttt{fork}\ (\texttt{thread}[\mathit{Th}']).\mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$	$= (\nu s{:}\ \mathsf{Amb}[\mathit{Synch}])\ ()^s\,\langle\!\langle\, \mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$
	$\mid\ (\nu t'{:}\ \mathsf{Amb}[\mathsf{shh}])t'[\mathsf{out}\ t.(\,\langle\!\langle\, \mathit{Th}' \,\rangle\!\rangle^{n,t'}_\Gamma \mid s[\mathsf{out}\ t'.\mathsf{in}\ t.\langle\rangle\,]\,)\,]$
$\langle\!\langle\, \texttt{fork}\ (\mathit{Cro}_1 \mid \mathit{Cro}_2).\mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$	$= \langle\!\langle\, \texttt{fork}\ (\mathit{Cro}_1).\texttt{fork}\ (\mathit{Cro}_2).\mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$
$\langle\!\langle\, \texttt{fork}\ ((\nu c{:}\mathit{Ch}(W))\mathit{Cro}).\mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$	$= (\nu c{:}\,\langle\!\langle\, \mathit{Ch}(W) \,\rangle\!\rangle\,)\,\langle\!\langle\, \texttt{fork}\ (\mathit{Cro}).\mathit{Th} \,\rangle\!\rangle^{n,t}_{\Gamma,c:W}$
$\langle\!\langle\, \texttt{fork}\ ((\nu m{:}\mathit{Node})\mathit{Cro}).\mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$	$= (\nu m{:}\,\langle\!\langle\, \mathit{Node} \,\rangle\!\rangle\,)\,\langle\!\langle\, \texttt{fork}\ (\mathit{Cro}).\mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$
$\langle\!\langle\, \texttt{spawn}\ m[\mathit{Cro}].\mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$	$= (\nu s{:}\ \mathsf{Amb}[\mathit{Synch}])\ ()^s\,\langle\!\langle\, \mathit{Th} \,\rangle\!\rangle^{n,t}_\Gamma$
	$\mid\ m[\mathsf{out}\ t.\mathsf{out}\ n.(\,\langle\!\langle\, \mathit{Cro} \,\rangle\!\rangle^m_\Gamma \mid s[\mathsf{out}\ m.\mathsf{in}\ n.\mathsf{in}\ t.\langle\rangle\,]\,)\,]$

The encoding of a network is parametric in a type environment Γ that we use to record the types of the values transported by channels (we need this to implement channels and their operations, as the parameter of an input channel is not typed in the source calculus). The encoding of a crowd is parametrized also by the current node n of the crowd; the encoding of thread expressions has as further parameter the name of the ambient that encloses the (translation of) the thread.

Nodes are represented as silent ambients whose sub-ambients encode either channels or threads. Threads also are enclosed into silent ambients (with fresh names), and thread migration is obtained by having threads exit the current node and enter the destination node. A few additional remarks are in order for the encoding of threads. First, in the encoding of fork and spawn we need a synchronization ambient s to trigger the continuation of the thread. Second, the encoding of forks must be given by cases on the concerned crowd: since we do not have open (as in [Cardelli et al. 2000]) we cannot make a whole crowd exit the thread and unleash it in the node; instead, we make each single component of the crowd exit the thread individually. Finally, the encoding of input/output on channels needs fresh pilot ambients, named w for writer and r for reader, which exit the current thread, enter the channel at issue and synchronize (and, in the case of readers, bring the message back). As in Section 4.3, we need the moded typing system to type-check these pilot ambients. Readers and writers are not upward silent and still need to move across nodes, which are locally silent. Nevertheless, they are well-typed because, by moded typing, one can infer that their upward exchanges become active only when they are inside (an ambient encoding) a channel (with local exchanges of the right type).

Below, we sketch the translation of the daemon program:

$$n[\ \ rcv[\,!(x:R)\langle x\rangle\,]\ \mid (\nu c : Ch\langle\!\langle W\rangle\!\rangle)(\nu t : \mathsf{Amb}[\mathsf{shh}]))t[\,\langle\!\langle rcv(x).\ \_theRest\_\rangle\!\rangle\,]\ \ ]$$

where $R = \mathsf{Amb}[\mathsf{Amb}[\mathsf{shh}]]$, the encoding of the type of rcv. The (translation of the) outer thread, enclosed in the ambient t, starts by spawning a pilot ambient r that exits t and goes to the ambient rcv to fetch the input which triggers the continuation of "the rest" of the thread.

$$\ldots\ t[\,(\nu r\colon \mathsf{Amb}^{\circ}[R,R,\mathsf{r}])\ (x{:}R)^{r}\langle\!\langle\ \_theRest\_\rangle\!\rangle\ \mid r[\,\mathsf{out}\ t.\mathsf{in}\ rcv.(x{:}R)^{\uparrow}.\mathsf{out}\ rcv.\mathsf{in}\ t.\langle x\rangle\,]\,]\ \ldots$$

Again, note that typing the reader r as a pilot ambient is needed for the move of r out of t to type-check, even though t is upward silent.

10.1 Access Control Policies

We now discuss how to specify, and statically enforce, different access control policies in the distributed language. We do this by way of the encoding. In particular, we study various ways for introducing security levels in the source language, and for each of them, we define a corresponding encoding into the access control type system of Section 9. Then, we study how the access control policies induced by our type system translate back into corresponding policies in the source language.

The first policy results from associating channel types and threads with security levels, thus interpreting threads and channels as subjects and objects, respectively. The translation is easily adapted to this case. Let γ be a Γ-consistent security assignment (see Section 9.3), then:

$$\begin{array}{lcl}
\langle\!\langle \sigma Ch(W)\rangle\!\rangle & = & \sigma\mathsf{Amb}[\,\langle\!\langle W\rangle\!\rangle\,] \\
\langle\!\langle \sigma\mathtt{thread}[Th]\rangle\!\rangle^{n}_{\Gamma} & = & (\nu t\colon \sigma\mathsf{Amb}[\mathsf{shh}])\ldots \\
\langle\!\langle \bar{c}(x)\rangle\!\rangle^{n,t}_{\Gamma} & = & (\nu w\colon \gamma(t)\mathsf{Amb}^{\circ}[\mathsf{shh},\langle\!\langle \Gamma(c)\rangle\!\rangle,\mathsf{w}])\ldots \\
\langle\!\langle c(x).Th\rangle\!\rangle^{n,t}_{\Gamma} & = & (\nu r\colon \gamma(t)\mathsf{Amb}^{\circ}[\,\langle\!\langle \Gamma(c)\rangle\!\rangle,\langle\!\langle \Gamma(c)\rangle\!\rangle,\mathsf{r}])\ldots \\
\langle\!\langle \mathtt{fork}\ (Th').Th\rangle\!\rangle^{n,t}_{\Gamma} & = & (\nu s\colon \mathsf{Amb}[Synch])(\nu t'\colon \gamma(t)\mathsf{Amb}[\mathsf{shh}])\ldots
\end{array}$$

Here, commercial and military security for the result of the translation correspond directly to commercial and military security for the source terms. Back to our example, if we take the channel *rcv* to be high (i.e. $rcv{:}\top Ch(Node)$) and the outer thread of n to be low (i.e. `node` $n[\ldots\mathtt{thread}\bot[rcv(x)\ldots]])$, then military and commercial security policies would both reject the the attempt by the thread to read on *rcv* as a read-up. This results from the failure to type-check the the pilot ambient r, which must be given low-clearance, since the thread t has low clearance:

$$(\nu r\colon \bot\mathsf{Amb}^{\circ}[R,R,\mathsf{r}])\ (x{:}R)^{r}\langle\!\langle\ \_theRest\_\rangle\!\rangle \mid r[\,\mathsf{out}\ t.\mathsf{in}\ rcv.(x{:}R)^{\uparrow}.\mathsf{out}\ rcv.\mathsf{in}\ t.\langle x\rangle\,]$$

The culprit is the sub-term $\mathsf{in}\ rcv.(x{:}R)^{\uparrow}$. By trying to construct a typing derivation for the term $r[\,\ldots]$, we discover that we need to type-check $\mathsf{in}\ rcv.(x{:}R)^{\uparrow}$ at level $\bot$, i.e. we need a derivation for

$$rcv : \top\,\langle\!\langle\, Ch(Node)\,\rangle\!\rangle \vdash_{\bot} \mathsf{in}\ rcv.(x{:}R)^{\uparrow} : \mathsf{Pro}[\_,\_,\mathsf{r}]$$

To derive this judgement, we must apply the rule (PREFIX Δ) because the process will be upward-active right after the move. But then we need a derivation for the judgement $rcv : \top\,\langle\!\langle\, Ch(Node)\,\rangle\!\rangle \vdash_{\bot} \mathsf{in}\ rcv : \bot\mathsf{Cap}[\_,\mathsf{r}]$, which instead fails because the clearance $\top$ of *rcv* is incompatible with the clearance $\bot$ of the capability: this judgement is not derivable under any "no read-up" policy.

An alternative policy is to endow node types with security levels. Accordingly, threads receive the security level of the node in which they are created. This is a common situation in practice, where it is usually implemented in the form of a partition of the nodes in trusted and distrusted (i.e., with just two security levels). We can modify the encoding as follows:

$$\begin{aligned} \langle\!\langle\, \sigma Node\,\rangle\!\rangle &= \sigma\mathsf{Amb}[\mathsf{shh}] \\ \langle\!\langle\, \mathtt{thread}[Th]\,\rangle\!\rangle^{n}_{\Gamma} &= (\nu t\colon \gamma(n)\mathsf{Amb}[\mathsf{shh}])\ldots \end{aligned}$$

Note that with this policy a thread can move into a node of any level. However, once there, it can access only resources (channels) which are compatible with its own level. In our example, if we give the node n security level higher than node m, then the thread forked in m can still move into n but it will be allowed to write *rcv* only if we declared the channel *rcv* to be of a level compatible with the security policy in use and the level of m.

If we want to forbid threads to move into "incompatible" nodes (this is usually needed in two cases: when we partition nodes in reliable and not reliable and want sensible threads to be executed only on reliable nodes, or when we partition threads in trusted and distrusted and we want sensible nodes to execute only trusted threads) we can modify the translation so that moving threads notify their entrance to the node they enter:

$$\begin{aligned} \langle\!\langle\, \sigma Node\,\rangle\!\rangle &= \sigma\mathsf{Amb}[Synch] \\ \langle\!\langle\, \mathtt{node}\ n[Cro]\,\rangle\!\rangle_{\Gamma} &= n[\ \langle\!\langle\, Cro\,\rangle\!\rangle^{n}_{\Gamma} \mid !()\,] \\ \langle\!\langle\, \mathtt{thread}[Th]\,\rangle\!\rangle^{n}_{\Gamma} &= (\nu t\colon \gamma(n)\mathsf{Amb}[\mathsf{shh},Synch,\mathsf{w}])\ldots \\ \langle\!\langle\, \mathtt{fork}\ (Th').Th\,\rangle\!\rangle^{n,t}_{\Gamma} &= (\nu s\colon \mathsf{Amb}[Synch])(\nu t'\colon \gamma(t)\mathsf{Amb}[\mathsf{shh},Synch,\mathsf{w}])\ldots \\ \langle\!\langle\, \mathtt{go}\ m.Th\,\rangle\!\rangle^{n,t}_{\Gamma} &= \mathsf{out}\ n.\mathsf{in}\ m.\langle\rangle^{\uparrow}\,\langle\!\langle\, Th\,\rangle\!\rangle^{n,t}_{\Gamma} \end{aligned}$$

Any attempt by a thread to move into a non compatible node makes the thread ill-typed. In particular the use of commercial security in the result of the translation, corresponds in the distributed language to ensure a node-protection policy since a node will run only threads of security level higher than or equal to its level. This means that a node runs only threads coming from nodes it trusts. Instead the use of military security enforces a thread-protection policy since a thread will run only on nodes with security level higher than or equal to its level. This means that sensitive threads will be run only on reliable nodes. We leave the task to verify that the translation enforces the intended policies to the interested reader.

11. RELATED WORK

Besides Mobile Ambients and Seals, whose relationships with Boxed Ambients have been discussed all along, our approach shares motivations, and is superficially similar to Sewell and Vitek's Box-π [Sewell and Vitek 2000]. The technical development, however, is entirely different. We do not provide direct mechanisms for constructing *wrappers*, rather we propose new constructs for ambient interaction in the attempt to provide easier-to-monitor communications. Also, our form of communication is anonymous, and based on a notion of locality which is absent in the Box-π Calculus. Finally Box-π does not consider mobility which is a fundamental component of this work.

In [Hennessy and Riely 2002b] Hennessy and Riely develop a type system for access control in the Dπ-calculus, a distributed variant of π-calculus where processes are located, and may migrate across locations. In Dπ, communication occurs via named channels that are associated with read and write capabilities: the type system controls that each process reading or writing on a channel possesses the appropriate capability. A similar technique is adopted by De Nicola *et. al.* in [De Nicola et al. 2000] for KLAIM, a distributed language based on a variant of Linda with multiple "tuple spaces". The main difference with our approach lies in the topological structure of Dπ (and KLAIM) locations and Boxed Ambient processes. In Dπ the topology of locations is completely flat, while in BA ambients may be nested at will: the interplay between the dynamic nesting structure determined by moves, and the dynamic binding of the parent location $\uparrow$ for upward communication makes access control for BA more complex. In [Riely and Hennessy 1999], the type system for Dπ is extended to cope with *partially typed networks*, in which some of the agents (and/or locations) are untyped, hence untrusted: type safety for such networks requires a form of dynamic type checking. We discuss our plans towards such extension in Section 12.

Our approach is also related to the work by Hennessy and Riely on the *security π-calculus* [Hennessy and Riely 2002a], a variant of the π-calculus in which processes are syntactically defined as running at a given security level. In BA, instead, we assume that the security levels are specified by types, and the clearance associated with an ambient-type represents the clearance of resources and the process contained in ambients with that ambient. Besides access control, in [Hennessy and Riely 2002a] the authors also conduct an analysis of information flow, and develop a type system that provides static guarantees of *non-interference*, defined in terms of testing equivalence. Our current access control type system does not provide such guarantees: it only ensures the static detection of access violations, and of certain forms of implicit flows via hidden channels. We investigate a-type based analysis of information flow in Boxed Ambients in a companion paper [Crafa et al. 2002], where we develop static type system for non interference.

Our type systems for BA are clearly related to other work on control and data flow analysis [Bugliesi and Castagna 2001; Nielson et al. 1999; Nielson and Nielson 2000], and typing system for Mobile Ambients. In [Cardelli and Gordon 1999b] types guarantees absence of type confusion for communications. The type systems of [Cardelli et al. 1999] and [Zimmer 2000] provide control over ambients moves and opening. Furthermore, the introduction of *group* names [Cardelli et al. 2000] and the possibility of creating fresh groups, give flexible ways to statically prevent unwanted propagation of names. The powerful type discipline for Safe Ambients, presented in [Levi and Sangiorgi 2000], adds finer control over ambient interactions and removes all *grave interference*, resulting from non-deterministic choices between logically incompatible interactions. All those approaches are orthogonal to the access control issue we studied. We believe that similar typing disciplines as well as the use of group names, can be adapted to Boxed Ambients to obtain similar strong results. A paper more directly related to ours is [Dezani-Ciancaglini and Salvo 2000], where ambient types were associated with a security level, having a crucial role in secure interactions. The difference is that in [Dezani-Ciancaglini and Salvo 2000] the security checks are performed upon ambient opening and moves, while in our work we focus on read and write operations.

A final mention goes to Merro and Sassone's recent paper on Boxed Ambients [Merro and Sassone 2002], in which they provide BA with a novel and powerful type system that combines a rich notion of value subtyping with mobility types. The former is based on read/write exchange types, the latter draws on the notion of ambient groups from [Cardelli et al. 2000]. In addition, they study the use of co-capabilities in BA as a means to express explicit permission to access ambients. As noted by the authors, their typing technique appears orthogonal, hence fully compatible with our system of moded types. We have not yet investigated the possibility of integrating the two system by incorporating our access control types. Plan for future include work in that direction.

12. CONCLUSIONS

We have presented BA, a novel ambient-based process calculus in which ambients cannot be opened, and new primitives provide for a controlled form of value exchange across ambient boundaries, between parent and child. The design of the calculus is motivated by security concerns. Removing the open capability ensures that the code of untrusted agents will never mingle freely within trusted ambients and hence reduces the potential of security threats. The new communication primitives, in turn, allow more concise encodings of several programming examples, while at the same time providing more effective means for access control. We have developed two semantics for the calculus, and studied their inter-relationships as well as their respective relationship with Mobile Ambients. Arguably, the synchronous semantics is not adequate for distributed computations. Nevertheless, it is useful as it motivates the design of the moded typing system which, in turn, provides insight into how the asynchronous reductions can be combined with a flexible typing of communication and ambient mobility.

We have complemented the definition of the calculus with a study of different type systems. In particular, we have developed a sound type system for access control in multilevel security, that combines static guarantees of safety for process interaction, with the static detection of any malicious or accidental violations of the intended security policy.

There are several aspects relative to the type theory of BA, and its computational properties that deserve to be explored. We conclude our presentation by discussing some of these aspects below.

12.1 Typing

As other ambient-based calculi, BA suffers from a certain lack of flexibility in the typing of value exchange. In particular, the type of a boxed ambient is the union of all of the potential interactions that ambient may engage. Some forms of dynamic typing could be appealed to address this limitation. On the other hand, one should notice that boxed ambients have communication capabilities richer than those advertised in their types. In particular, a boxed ambient may use sub-ambients to hold interactions of different types using downward communication. In fact, we have shown that channels can be encoded in the calculus: in the implementation of a programming language based on BA, channels would clearly need to be made primitive, as done in the Seal calculus, and its implementations [Bryce and Vitek 2001].

A possible extension of our type system would be to enrich the current structure of ambient types with a further component for downward communication. This would be useful, for instance, to limit the power of an enclosing ambient on its sub-ambients. We believe this to be a viable option, that could be incorporated in our type system with no fundamental difficulty. On the other hand, this usage of types contrasts with our current interpretation of ambient types as interfaces, which describe what the context sees of an ambient. In addition, an ambient can protect itself from its enclosing context by relying on term-level constructs (by wrapping ambients or hiding their names).

A more serious limitation of the current access control type system of BA is that it assumes a centralized form of typing, where all the components of a system are type-checked under the same 'global' assumptions on the types of names. We are currently investigating two ways to overcome this limitation. One solution is to rely on forms of dynamic typing, as it is done in other type systems for Mobile (Safe) Ambients [Bugliesi and Castagna 2001], and other calculi [Riely and Hennessy 1999; De Nicola et al. 2000]. Specifically, as in [Bugliesi and Castagna 2001], the idea is to define a typed variant of BA in which each ambient carries a type environment, to be used for static, and *local* type checking. To ensure type soundness, static typing must then be complemented by a form of just-in-time type checking taking place an ambient crosses a boundary, to ensure the consistency of the local type assumptions of the moving and the target ambients.

An alternative solution is to introduce new primitives, based on cryptography, to protect trusted (i.e. typed) migrating agents against the untrusted sites they traverse, and to rely on a type system that separates trusted and untrusted and code, while allowing safe interactions with untrusted sites. Work in this direction has been initiated in [Bugliesi et al. 2002]

12.2 Semantics and implementation

The implementation of BA poses some new interesting problems. On one side, the absence of open simplifies the problem with respect to MA. In current implementations of Mobile Ambients [Sangiorgi and A. 2001; Fournet et al. 2000], the opening of an ambient is a rather complex operation that transforms the opened ambient into a *forwarder* for all the synchronization requests from the ambient's parent and children. No such mechanism is required for BA.

On the other side, the semantics of communication poses new implementation challenges, which result from the inherent nondeterminism in the synchronization of the local input/output operation in BA. In particular, with the current reductions (both synchronous and asynchronous), an output $\langle M \rangle$ may nondeterministically synchronize with an input from from the parent, or from a child, or from a (local) sibling process. This semantics has a strong motivation in our design, namely it enforces the interpretation of anonymous channels as resources located *inside* ambients: indeed, our very definition of access control relies on this interpretation.

One could devise different reductions, such as those defined below.

$$\begin{array}{rcl} (x)^n P \mid n[\,\langle M \rangle^{\uparrow} Q \mid R\,] & \longrightarrow & P\{x := M\} \mid n[\,Q \mid R\,] \\ \langle M \rangle^n P \mid n[\,(x)^{\uparrow} Q \mid R\,] & \longrightarrow & P \mid n[\,Q\{x := M\} \mid R\,] \end{array}$$

These reduction yield a semantics which is similar to that adopted in [Castagna et al. 2001] for the Seal Calculus, and is based on the idea that each ambient comes equipped with two mutually non-interfering channels, respectively for local and upward communications. Hierarchical communication, whose rules are shown above, is indicated by a pair of distinct constructors, simultaneously on input and output, so that no communication interference is possible. The upward channel can be thought of as a gateway between parent and child, located at the child's and traveling with it, and poses no particular implementation challenges.

One problem with adopting this semantics for BA, is that it results in a poorly expressive calculus. For instance, it would not be possible to encode the form of message broadcasting implemented by the following term: $a[\,!\,\langle M \rangle\,]$. Here a is as an "information site" which any ambient can enter to get a copy of M (reading it from upward, after having entered a). The same protocol could hardly be expressed with the reductions given above, as they require an ambient to know the names of its children in order to communicate with them. We can recover expressiveness, as suggested in [Bugliesi et al. 2002], by introducing co-actions of the form $\mathsf{coin}(x)$ having the effect of binding the name of an incoming ambient to the variable x. Using this form of co-actions, we can program the information site as follows: $a[\,!\,\mathsf{coin}(x).\langle M \rangle^x\,]$, and have clients be coded as $c[\,\mathsf{in}\ a(x)^{\uparrow} P\,]$.

When it comes to access control, however, this encoding is problematic as it exchanges the roles of readers and writers. In the initial example it is the client that reads from the server; in its coding, it is the server that writes to the client. Thus, while the new reductions would simplify the implementation, they would also undermine our access control framework.

A solution we are currently investigating is to adopt the asynchronous semantics, as defined in Section 8, and implement it with reductions such as those displayed above. A similar study has been conducted for the synchronous semantics of BA in the recent paper [Bugliesi et al. 2002]. That paper provides a partial solution based on the use of a (mixed) guarded-choice operator, which in turn requires a complex (cf. [Nestmann 2000]) form distributed consensus to capture the desired synchronizations. It appears that a more satisfactory implementation is possible for the asynchronous semantics of BA, as in that we could rely on a more treatable form of choice, based on an input-choice operator. Our plans of future research include work in that direction.

Acknowledgments

The stimulating cricitism and valuable suggestions by the anonymous referees of TOPLAS were of great help in improving the presentation. We also acknowledge the comments from the program committees of TACS'01, CONCUR'01, and CONCOORD'01.

This work was partially supported by 'MyThS: Models and Types for Security in Mobile Distributed Systems', EU FET-GC IST-2001-32617 and by MIUR Project "Modelli formali per la sicurezza (MEFISTO)".

REFERENCES

AMADIO, R. AND PRASAD, S. 1994. Localities and failures. In *FST&TCS*. Number 880 in Lecture Notes in Computer Science. 206–216.

BELL, D. AND PADULA, L. L. 1976. Secure computer system: Unified exposition and multics interpretation,. Tech. Rep. MTR-2997, MITRE Corporation, Bedford, MA 01730. Mar.

BOUDOL, G. 1992. Asynchrony and the π-calculus. Research Report 1702, INRIA, http://www.inria.fr/rrrt/rr-1702.html. Also available from http://www-sop.inria.fr/mimosa/personnel/Gerard.Boudol.html.

BRYCE, C. AND VITEK, J. 2001. The JavaSeal mobile agent kernel. *Autonomous Agents and Multi-Agent Systems 4,* 4, 359–384.

BUGLIESI, M. AND CASTAGNA, G. 2001. Secure safe ambients. In *Proc. of POPL'01*. ACM Press, 222–235.

BUGLIESI, M., CASTAGNA, G., AND CRAFA, S. 2001a. Boxed ambients. In *TACS'01*. Number 2215 in Lecture Notes in Computer Science. Springer-Verlag, 38–63.

BUGLIESI, M., CASTAGNA, G., AND CRAFA, S. 2001b. Reasoning about security in Mobile Ambients. In *CONCUR'01*. Number 2154 in Lecture Notes in Computer Science. Springer-Verlag, 102–120.

BUGLIESI, M., CRAFA, S., MERRO, M., AND SASSONE, V. To appear. 2002. Communication interference in mobile boxed ambients. In *FSTTCS'02: Int. Conf. on Foundations of Software Technology and Theoretical Computer Science*.

BUGLIESI, M., CRAFA, S., PRELIC, A., AND SASSONE, V. 2002. Cryptographic Primitives in Mobile Boxed Ambients. Submitted.

CARDELLI, L. 1999. *Abstractions for Mobile Computation*. Lecture Notes in Computer Science, vol. 1603. Springer-Verlag, 51–94.

CARDELLI, L., GHELLI, G., AND GORDON, A. 1999. Mobility types for Mobile Ambients. In *Proceedings of ICALP '99*. Number 1644 in Lecture Notes in Computer Science. Springer-Verlag, 230–239.

CARDELLI, L., GHELLI, G., AND GORDON, A. D. 2000. Ambient groups and mobility types. In *International Conference IFIP TCS*. Number 1872 in Lecture Notes in Computer Science. Springer-Verlag, 333–347.

CARDELLI, L. AND GORDON, A. 1998. Mobile Ambients. In *Proceedings of F0SSaCS'98*. Number 1378 in Lecture Notes in Computer Science. Springer-Verlag, 140–155.

CARDELLI, L. AND GORDON, A. 1999a. Equational properties for Mobile Ambients. In *Proceedings FoSSaCS '99*. Springer-Verlag LNCS.

CARDELLI, L. AND GORDON, A. 1999b. Types for Mobile Ambients. In *Proceedings of POPL '99*. ACM Press, 79–92.

CASTAGNA, G., GHELLI, G., AND ZAPPA NARDELLI, F. 2001. Typing mobility in the Seal Calculus. In *CONCUR'01*. Number 2154 in Lecture Notes in Computer Science. Springer-Verlag, 82–101.

CRAFA, S., BUGLIESI, M., AND CASTAGNA, G. 2002. Information flow security for boxed ambients. In *F-WAN: Int. Workshop on Foundations of Wide Area Networks*. Number 66(3) in ENTCS.

DE NICOLA, R., FERRARI, G., AND PUGLIESE, R. 1998. KLAIM: A Kernel Language for Agents Interaction and Mobility. *IEEE Transactions on Software Engineering 24,* 5, 315–330.

DE NICOLA, R., FERRARI, G., AND PUGLIESE, R. 2000. Programming Access Control: the KLAIM Experience. In *Proceedings of CONCUR'00*. Number 1877 in Lecture Notes in Computer Science. Springer-Verlag, 48–65.

DE NICOLA, R., FERRARI, G., PUGLIESE, R., AND VENNERI, B. 2000. Types for access control. *Theoretical Computer Science 240,* 1, 215–254.

DEGANO, P., LEVI, F., AND BODEI, C. 2000. Safe ambients: Control flow analysis and security. In *Proceedins of ASIAN '00*. LNCS, vol. 1961. Springer-Verlag, 199–214.

DEPARTMENT OF DEFENSE, U. 1985. DoD trusted computer system evaluation criteria, (the orange book). DOD 5200.28-STD.

DEZANI-CIANCAGLINI, M. AND SALVO, I. 2000. Security types for Safe Mobile Ambients. In *Proceedings of ASIAN '00*. Springer-Verlag, 215–236.

FOCARDI, R. AND GORRIERI, R. 1997. Non interference: Past, present and future. In *Proceedings of DARPA Workshop on Foundations for Secure Mobile Code*. 26–28.

FOURNET, C., GONTHIER, G., LÉVY, J.-J., MARANGET, L., AND RÉMY, D. 1996. A calculus of mobile agents. In *7th International Conference on Concurrency Theory (CONCUR '96)*. Lecture Notes in Computer Science, vol. 1119. Springer-Verlag, 406–421.

FOURNET, C., LEVY, J.-J., AND A, S. 2000. An asynchronous, distributed implementation of mobile ambients. In *International Conference IFIP TCS*. Number 1872 in Lecture Notes in Computer Science. Springer-Verlag.

GOGUEN, J. AND MESEGUER, J. 1982. Security policy and security models. In *Proceedings of Symposium on Secrecy and Privacy*. IEEE Computer Society, 11–20.

GOLLMANN, D. 1999. *Computer Security*. John Wiley & Sons Ltd.

HENNESSY, M. AND RIELY, J. 2002a. Information flow vs resource access in the asynchronous π-calculus. *ACM Trans. Program. Lang. Syst. 24,* 5, 566–591.

HENNESSY, M. AND RIELY, J. 2002b. Resource access control in systems of mobile agents. *Information and Computation 173*, 82–120.

LEVI, F. AND SANGIORGI, D. 2000. Controlling interference in Ambients. In *POPL '00*. ACM Press, 352–364.

MERRO, M. AND HENNESSY, M. 2002. Bisimulation conguences in Safe Ambients. In *POPL'02*. ACM Press, 71–80.

MERRO, M. AND SASSONE, V. 2002. Typing and subtyping mobility in boxed ambients. In *Proceedings of Concur'02*. Lecture Notes in Computer Science. Springer-Verlag, 304 – 320.

NESTMANN, U. 2000. What is a 'Good' Encoding of Guarded Choice? *Information and Computation 156*, 287–319.

NIELSON, F., NIELSON, H. R., HANSEN, R. R., AND JENSEN, J. G. 1999. Validating firewalls in mobile ambients. In *Proc. CONCUR '99*. Number 1664 in Lecture Notes in Computer Science. Springer-Verlag, 463–477.

NIELSON, H. R. AND NIELSON, F. 2000. Shape analysis for mobile ambients. In *POPL '00*. ACM Press, 142–154.

RIELY, J. AND HENNESSY, M. 1998. A typed language for distributed mobile processes. In *Proceedings of POPL'98*. ACM Press, 378–390.

RIELY, J. AND HENNESSY, M. 1999. Trust and partial typing in open systems of mobile agents. In *Proceedings of POPL '99*. ACM Press, 93–104.

SANGIORGI, D. AND A., V. 2001. A distributed abstract mathine for Safe Ambients. In *Proc. of ICALP'01*. Lecture Notes in Computer Science, vol. 2076. Springer-Verlag, 408–420.

SEWELL, P. AND VITEK, J. 2000. Secure composition of untrusted code: Wrappers and causality types. In *13th IEEE Computer Security Foundations Workshop*. To Appear in *Journal of Computer Security*.

VITEK, J. AND CASTAGNA, G. 1999. Seal: A framework for secure mobile computations. In *Internet Programming Languages*. Number 1686 in Lecture Notes in Computer Science. Springer-Verlag, 47–77.

ZIMMER, P. 2000. Subtyping and typing algorithms for mobile ambients. In *Proceedins of FoSSaCS '99*. Lecture Notes in Computer Science, vol. 1784. Springer-Verlag, 375–390.

A. TYPING RULES

We list the complete set of rules for the type systems described in Sections 5, 7, and 9. In order to have a more compact set of rules, we use ${}^{\mu}F$ to denote any of the exchanges ${}^{\vartriangle}F, {}^{\bullet}F, {}^{\circ}F$, and use ${}^{?}F$ to denote either ${}^{\mu}F$ or F. Similarly we use $\mathsf{Amb}^{?}[E,F,\mathcal{A}]$ to denote either $\mathsf{Amb}[E,F,\mathcal{A}]$ or $\mathsf{Amb}^{\circ}[E,F,\mathcal{A}]$. The use of such shorthands make it possible to express the rules of § 5, § 7 as instances of the rules listed below: specifically, the type system of § 7 derives by erasing security levels and access modes, and the system of § 5 from further erasing all rules that involve non-empty modes on the types, and moded judgements.

A.1 Good Environments and Expressions

(EMPTY)
$$\frac{}{\varnothing \vdash \diamond}$$

(VAR)
$$\frac{\Gamma \vdash \diamond \quad x \notin Dom(\Gamma)}{\Gamma, x : W \vdash \diamond}$$

(NAME)
$$\frac{\Gamma \vdash \diamond \quad n \notin Dom(\Gamma)}{\Gamma, n : W \vdash \diamond}$$

(PROJECTION)
$$\frac{\Gamma(M) = W \quad \Gamma \vdash \diamond}{\Gamma \vdash M : W}$$

(IN)
$$\frac{\Gamma \vdash M : \rho\mathsf{Amb}^{?}[E,F,\mathcal{B}] \quad \mathcal{P}(\sigma,\rho,\mathcal{A}) \quad G \leqslant E}{\Gamma \vdash \mathsf{in}\ M : \sigma\mathsf{Cap}[G,\mathcal{A}]}$$

(OUT)
$$\frac{\Gamma \vdash M : \sigma\mathsf{Amb}[E,F,\mathcal{B}] \quad G \leqslant F,\ \mathcal{A} \leqslant \mathcal{B}}{\Gamma \vdash \mathsf{out}\ M : \rho\mathsf{Cap}[G,\mathcal{A}]}$$

(PATH)
$$\frac{\Gamma \vdash M_1 : \sigma\mathsf{Cap}[E,\mathcal{A}] \quad \Gamma \vdash M_2 : \sigma\mathsf{Cap}[E,\mathcal{A}]}{\Gamma \vdash M_1.M_2 : \sigma\mathsf{Cap}[E,\mathcal{A}]}$$

(OUT ∘)
$$\frac{\Gamma \vdash M : \sigma\mathsf{Amb}^{\circ}[E,F,\mathcal{B}]}{\Gamma \vdash \mathsf{out}\ M : \rho\mathsf{Cap}[\mathsf{shh},\mathcal{A}]}$$

(POLYPATH)
$$\frac{\Gamma \vdash^{\circ} M_1 : \rho\mathsf{Cap}[F,\mathcal{A}] \quad \Gamma \vdash^{\circ} M_2 : \sigma\mathsf{Cap}[E,\mathcal{B}]}{\Gamma \vdash^{\circ} M_1.M_2 : \sigma\mathsf{Cap}[E,\mathcal{B}]}$$

(POLYCAP)
$$\frac{\Gamma \vdash M : \sigma\mathsf{Cap}[E,\mathcal{A}]}{\Gamma \vdash^{\circ} M : \sigma\mathsf{Cap}[E,\mathcal{A}]}$$

A.2 Good Processes

(PREFIX)
$$\frac{\Gamma \vdash M : \sigma\mathsf{Cap}[F,\mathcal{A}] \quad \Gamma \vdash_{\sigma} P : \mathsf{Pro}[E,F,\mathcal{A}]}{\Gamma \vdash_{\sigma} M.P : \mathsf{Pro}[E,F,\mathcal{A}]}$$

(PREFIX ∘)
$$\frac{\Gamma \vdash^{\circ} M : \rho\mathsf{Cap}[G,\mathcal{B}] \quad \Gamma \vdash_{\sigma} P : \mathsf{Pro}[E,{}^{\circ}F,\mathcal{A}]}{\Gamma \vdash_{\sigma} M.P : \mathsf{Pro}[E,{}^{\circ}F,\mathcal{A}]}$$

(PREFIX △)
$$\frac{\Gamma \vdash^{\circ} M : \sigma\mathsf{Cap}[F,\mathcal{A}] \quad \Gamma \vdash_{\sigma} P : \mathsf{Pro}[E,{}^{\vartriangle}F,\mathcal{A}]}{\Gamma \vdash_{\sigma} M.P : \mathsf{Pro}[E,{}^{\circ}F,\mathcal{A}]}$$

(PREFIX •)
$$\frac{\Gamma \vdash M : \sigma\mathsf{Cap}[F,\mathcal{A}] \quad \Gamma \vdash_{\sigma} P : \mathsf{Pro}[E,{}^{\bullet}F,\mathcal{A}]}{\Gamma \vdash_{\sigma} M.P : \mathsf{Pro}[E,{}^{\bullet}F,\mathcal{A}]}$$

(PAR)
$$\frac{\Gamma \vdash_{\sigma} P : \mathsf{Pro}[E,F,\mathcal{A}] \quad \Gamma \vdash_{\sigma} Q : \mathsf{Pro}[E,F,\mathcal{A}]}{\Gamma \vdash_{\sigma} P \mid Q : \mathsf{Pro}[E,F,\mathcal{A}]}$$

(PAR μ)
$$\frac{\Gamma \vdash_{\sigma} P : \mathsf{Pro}[E,{}^{\mu}F,\mathcal{A}] \quad \Gamma \vdash_{\sigma} Q : \mathsf{Pro}[E,{}^{\bullet}F,\mathcal{A}]}{\Gamma \vdash_{\sigma} P \mid Q, Q \mid P : \mathsf{Pro}[E,{}^{\mu}F,\mathcal{A}]}$$

(DEAD)
$$\frac{\Gamma \vdash \diamond}{\Gamma \vdash_{\sigma} \mathbf{0} : T}$$

(NEW)
$$\frac{\Gamma, n : \rho\mathsf{Amb}^{?}[E,F] \vdash_{\sigma} P : T}{\Gamma \vdash_{\sigma} (\nu n{:}\rho\mathsf{Amb}^{?}[E,F])P : T}$$

(REPL$^{\bullet}$)
$$\frac{\Gamma \vdash_{\sigma} P : \mathsf{Pro}[E,{}^{\bullet}F,\mathcal{A}]}{\Gamma \vdash_{\sigma} !P : \mathsf{Pro}[E,{}^{\bullet}F,\mathcal{A}]}$$

(REPL)
$$\frac{\Gamma \vdash_{\sigma} P : \mathsf{Pro}[E,F,\mathcal{A}]}{\Gamma \vdash_{\sigma} !P : \mathsf{Pro}[E,F,\mathcal{A}]}$$

$$\textsc{(Amb)}\quad \frac{\Gamma \vdash a : \sigma\mathsf{Amb}[E,F,\mathcal{A}] \quad \Gamma \vdash_\sigma P : \mathsf{Pro}[E,F,\mathcal{A}] \quad \mathcal{P}(\sigma,\rho,\mathcal{A})}{\Gamma \vdash_\rho a[P] : \mathsf{Pro}[F,{}^\bullet H,\mathcal{B}]}$$

$$\textsc{(Subsumption)}\quad \frac{\Gamma \vdash_\sigma P : T \quad T \leqslant T'}{\Gamma \vdash_\sigma P : T'}$$

$$\textsc{(Amb }\vartriangle\textsc{)}\quad \frac{\Gamma \vdash a : \sigma\mathsf{Amb}^\circ[E,F,\mathcal{A}] \quad \Gamma \vdash_\sigma P : \mathsf{Pro}[E,{}^\vartriangle F,\mathcal{A}] \quad \mathcal{P}(\sigma,\rho,\mathcal{A})}{\Gamma \vdash_\rho a[P] : \mathsf{Pro}[F,{}^\bullet H,\mathcal{B}]}$$

$$\textsc{(Amb }\circ\textsc{)}\quad \frac{\Gamma \vdash a : \sigma\mathsf{Amb}^\circ[E,F,\mathcal{A}] \quad \Gamma \vdash_\sigma P : \mathsf{Pro}[E,{}^\circ F,\mathcal{A}]}{\Gamma \vdash_\rho a[P] : \mathsf{Pro}[G,{}^\bullet H,\mathcal{B}]}$$

In the input rules below, we make the usual assumption that if $\tilde{x} = x_1,\ldots x_k$ then $W = W_1 \times \cdots \times W_k$, and we use the notation $\Gamma,\tilde{x} : W$ as a shorthand for $\Gamma, x_1 : W_1, \ldots x_k : W_k$.

$$\textsc{(Input }\star\textsc{)}\quad \frac{\Gamma,\tilde{x} : W \vdash_\sigma P : \mathsf{Pro}[W,{}^? F,\mathcal{A}]}{\Gamma \vdash_\sigma (\tilde{x} : W)P : \mathsf{Pro}[W,{}^? F,\mathcal{A}]}$$

$$\textsc{(Output }\star\textsc{)}\quad \frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash_\sigma P : \mathsf{Pro}[W_1 \times \cdots \times W_k,{}^? F,\mathcal{A}]}{\Gamma \vdash_\sigma \langle M_1,\ldots,M_k\rangle P : \mathsf{Pro}[W_1 \times \cdots \times W_k,{}^? F,\mathcal{A}]}$$

$$\textsc{(Input }M\textsc{)}\quad \frac{\Gamma,\tilde{x} : W \vdash_\sigma P : \mathsf{Pro}[E,{}^\mu F,\mathcal{A}] \quad \Gamma \vdash M : \rho\mathsf{Amb}^?[W,G,\mathcal{B}] \quad \mathcal{P}(\sigma,\rho,\mathsf{r})}{\Gamma \vdash_\sigma (\tilde{x} : W)^M P : \mathsf{Pro}[E,{}^\mu F,\mathcal{A}]}$$

$$\textsc{(Output }M\textsc{)}\quad \frac{\Gamma \vdash N_i : W_i \quad \Gamma \vdash_\sigma P : \mathsf{Pro}[E,{}^\mu F,\mathcal{A}] \quad \Gamma \vdash M : \rho\mathsf{Amb}^?[W_1 \times \cdots \times W_k,G,\mathcal{B}] \quad \mathcal{P}(\sigma,\rho,\mathsf{w})}{\Gamma \vdash_\sigma \langle N_1,\ldots,N_k\rangle^M P : \mathsf{Pro}[E,{}^\mu F,\mathcal{A}]}$$

$$\textsc{(Input }\uparrow\textsc{)}\quad \frac{\Gamma,\tilde{x} : W \vdash_\sigma P : \mathsf{Pro}[F,W,\mathcal{A}] \quad \mathsf{r} \leqslant \mathcal{A}}{\Gamma \vdash_\sigma (\tilde{x} : W)^\uparrow P : \mathsf{Pro}[F,W,\mathcal{A}]}$$

$$\textsc{(Input }\uparrow\vartriangle\textsc{)}\quad \frac{\Gamma,\tilde{x} : W \vdash_\sigma P : \mathsf{Pro}[F,{}^\vartriangle W,\mathcal{A}] \quad \mathsf{r} \leqslant \mathcal{A}}{\Gamma \vdash_\sigma (\tilde{x} : W)^\uparrow P : \mathsf{Pro}[F,{}^\vartriangle W,\mathcal{A}]}$$

$$\textsc{(Output }\uparrow\textsc{)}\quad \frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash_\sigma P : \mathsf{Pro}[F,W_1 \times \cdots \times W_k,\mathcal{A}] \quad \mathsf{w} \leqslant \mathcal{A}}{\Gamma \vdash_\sigma \langle M_1,\ldots,M_k\rangle^\uparrow P : \mathsf{Pro}[F,W_1 \times \cdots \times W_k,\mathcal{A}]}$$

$$\textsc{(Output }\uparrow\vartriangle\textsc{)}\quad \frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash_\sigma P : \mathsf{Pro}[F,{}^\vartriangle(W_1 \times \cdots \times W_k),\mathcal{A}] \quad \mathsf{w} \leqslant \mathcal{A}}{\Gamma \vdash_\sigma \langle M_1,\ldots,M_k\rangle^\uparrow P : \mathsf{Pro}[F,{}^\vartriangle(W_1 \times \cdots \times W_k),\mathcal{A}]}$$

For the asynchronous calculus we need two additional rules for processes forms:

$$\textsc{(Asynch Output)}\quad \frac{\Gamma \vdash M_i : W_i}{\Gamma \vdash_\sigma \langle M_1,\ldots,M_k\rangle : \mathsf{Pro}[W_1 \times \cdots \times W_k,{}^\bullet F,\mathcal{A}]}$$

$$\frac{\Gamma \vdash N_i : W_i \quad \Gamma \vdash M : \rho\mathsf{Amb}^?[W_1 \times \cdots \times W_k, G, \mathcal{B}] \quad \mathcal{P}(\sigma, \rho, \mathsf{w})}{\Gamma \vdash_\sigma \langle N_1, \ldots, N_k \rangle^M : \mathsf{Pro}[E, {}^\bullet F, \mathcal{A}]} \ (\text{ASYNCH OUTPUT } M)$$

B. PROOFS OF SUBJECT REDUCTION AND TYPE SOUNDNESS

The existence of multiple typing rules for the same syntactic form causes a proliferation of typing derivations for the same judgment. The following lemma shows that we can focus attention on derivations of more regular shape without losing generality.

LEMMA TYPING OF PROCESSES.

Ambients. Assume $\Gamma \vdash_\rho a[P] : T$. Then there exist E and F exchanges, and $\mathcal{A}$ and $\mathcal{B}$ access modes, such that $\mathsf{Pro}[E, {}^\bullet F, \mathcal{B}] \leqslant T$ and $\Gamma \vdash_\rho a[P] : \mathsf{Pro}[E, {}^\bullet F, \mathcal{B}]$ is derivable from the following assumptions, where H and K are arbitrary exchanges:

($\mathbf{a_1}$) either $\Gamma \vdash a : \sigma\mathsf{Amb}[H, E, \mathcal{A}]$, and $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, E, \mathcal{A}]$ and $\mathcal{P}(\sigma, \rho, \mathcal{A})$

($\mathbf{a_2}$) or $\Gamma \vdash a : \sigma\mathsf{Amb}^\circ[H, E, \mathcal{A}]$, and $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^\vartriangle E, \mathcal{A}]$ and $\mathcal{P}(\sigma, \rho, \mathcal{A})$

($\mathbf{a_3}$) or $\Gamma \vdash a : \sigma\mathsf{Amb}^\circ[H, K, \mathcal{A}]$ and$\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^\circ K, \mathcal{A}]$

Parallel Composition. Assume $\Gamma \vdash_\sigma P \mid Q : T$. Then there exist H and E (exchanges) and $\mathcal{A}$ (access mode) such that $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[E, {}^? E, \mathcal{A}]$ with $\mathsf{Pro}[E, {}^? E, \mathcal{A}] \leqslant T$, and the last judgment is derivable from the following assumptions.

($\mathbf{c_1}$) if $T \leqslant \mathsf{Pro}[H, E, \mathcal{A}]$ then $\Gamma \vdash_\sigma P : T$ and $\Gamma \vdash_\sigma Q : T$

($\mathbf{c_2}$) if $T = \mathsf{Pro}[H, {}^\circ E, \mathcal{A}]$ then either $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^\bullet E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^\circ E, \mathcal{A}]$, or $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^\circ E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^\bullet E, \mathcal{A}]$

($\mathbf{c_3}$) if $T = \mathsf{Pro}[H, {}^\vartriangle E, \mathcal{A}]$ then either $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^\bullet E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^\vartriangle E, \mathcal{A}]$, or $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^\vartriangle E, \mathcal{A}]$ $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^\bullet E, \mathcal{A}]$, or else $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, E, \mathcal{A}]$

Prefix. Assume $\Gamma \vdash_\sigma M.P : T$. Then there exist H and E exchanges, and $\mathcal{A}$ access mode such that $\Gamma \vdash_\sigma M.P : \mathsf{Pro}[H, {}^? E, \mathcal{A}]$ with $\mathsf{Pro}[H, {}^? E, \mathcal{A}] \leqslant T$, and the last judgment is derivable from the following assumptions, where G is any exchange, $\mathcal{B}$ is any access mode, and ρ any security level:

($\mathbf{p_1}$) if $T \leqslant \mathsf{Pro}[H, E, \mathcal{A}]$ then $\Gamma \vdash M : \sigma\mathsf{Cap}[E, \mathcal{A}]$ and $\Gamma \vdash_\sigma P : T$

($\mathbf{p_2}$) if $\mathsf{Pro}[H, {}^\circ E, \mathcal{A}] \leqslant T$ then $\Gamma \vdash\!\circ M : \rho\mathsf{Cap}[G, \mathcal{B}]$ and $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^\circ E, \mathcal{A}]$, or $\Gamma \vdash\!\circ M : \sigma\mathsf{Cap}[E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^\vartriangle E, \mathcal{A}]$.

Input. Assume $\Gamma \vdash_\sigma (\tilde{x} : W)^\eta P : T$. Then there exist E, F, G, $\mathcal{A}$, and ρ such that:

($\mathbf{i_1}$) if $\eta = \star$ then $\Gamma, \tilde{x} : W \vdash_\sigma P : \mathsf{Pro}[W, {}^? E, \mathcal{A}] \leqslant T$

($\mathbf{i_2}$) if $\eta = M$ then $\Gamma, \tilde{x} : W \vdash_\sigma P : \mathsf{Pro}[E, {}^\mu F, \mathcal{A}] \leqslant T$, $\Gamma \vdash M : \rho\mathsf{Amb}^?[W, G, \mathcal{A}]$, and $\mathcal{P}(\sigma, \rho, \mathsf{r})$.

($\mathbf{i_3}$) if $\eta = \uparrow$ then $\Gamma, \tilde{x} : W \vdash_\sigma P : \mathsf{Pro}[E, W, \mathcal{A}] \leqslant T$, or $\Gamma, \tilde{x} : W \vdash_\sigma P : \mathsf{Pro}[E, {}^\vartriangle W, \mathcal{A}] \leqslant T$, with $\mathsf{r} \leqslant \mathcal{A}$

Output. Assume $\Gamma \vdash_\sigma \langle M_1, \ldots, M_k \rangle^\eta P : T$. Then there exist E, F, G, $W_1, \ldots, W_k$, $\mathcal{A}$, and ρ such that:

($\mathbf{o_1}$) if $\eta = \star$ then $\Gamma \vdash_\sigma P : \mathsf{Pro}[W_1 \times \cdots \times W_k, {}^{?}E, \mathcal{A}] \leqslant T$ and $\Gamma \vdash M_i : W_i$.

($\mathbf{o_2}$) if $\eta = N$ then $\Gamma \vdash_\sigma P : \mathsf{Pro}[E, {}^{\mu}F, \mathcal{A}] \leqslant T$, $\Gamma \vdash N : \rho\mathsf{Amb}^{?}[W_1 \times \cdots \times W_k, G, \mathcal{A}]$, and $\Gamma \vdash M_i : W_i$, with $\mathcal{P}(\sigma, \rho, \mathsf{w})$

($\mathbf{o_3}$) if $\eta = \uparrow$ then $\Gamma \vdash_\sigma P : \mathsf{Pro}[E, W_1 \times \cdots \times W_k, \mathcal{A}] \leqslant T$,
or $\Gamma \vdash_\sigma P : \mathsf{Pro}[E, {}^{\vartriangle}(W_1 \times \cdots \times W_k), \mathcal{A}] \leqslant T$, with $\Gamma \vdash M_i : W_i$ and $\mathsf{w} \leqslant \mathcal{A}$.

PROOF. We need to show that we have captured all the possible cases.

Ambients. The judgment $\Gamma \vdash_\rho a[P] : T$ must have been derived by an application of one of the (AMB) rules followed by any number of subsumption steps. An inspection of the typing rules for ambients proves the claim.

Parallel Composition. The first part of is obvious, the second part is proved as follows.

($\mathbf{c_1}$). $T \leqslant \mathsf{Pro}[H, E, \mathcal{A}]$ covers two cases: $T = \mathsf{Pro}[H, {}^{\bullet}E, \mathcal{A}]$ or $T = \mathsf{Pro}[H, E, \mathcal{A}]$. In the first case, $\Gamma \vdash_\sigma P \mid Q : T$ must be derived by (PAR $\bullet$) from $\Gamma \vdash_\sigma P : \mathsf{Pro}[H', {}^{\bullet}E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H', {}^{\bullet}E, \mathcal{A}]$, with $H' \leqslant H$, followed by one or more subsumption steps. Thus $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^{\bullet}E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^{\bullet}E, \mathcal{A}]$ are also derivable, and from these judgments one derives the $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[H, {}^{\bullet}E, \mathcal{A}]$ by (PAR $\bullet$).

In the second case, $\Gamma \vdash_\sigma P \mid Q : T$ must have been derived either from the judgements $\Gamma \vdash_\sigma P : \mathsf{Pro}[H', {}^{\bullet}E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H', {}^{\bullet}E, \mathcal{A}]$ by (PAR $\bullet$) followed by subsumption (with $H' \leqslant H$), or from the judgements $\Gamma \vdash_\sigma P : \mathsf{Pro}[H', E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H', E, \mathcal{A}]$, by (PAR) again followed by subsumption. In both cases $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, E, \mathcal{A}]$ are derivable. From these judgments one derives $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[H, E, \mathcal{A}]$ by (PAR).

($\mathbf{c_2}$). $\Gamma \vdash_\sigma P \mid Q : T$ may have been derived from $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[H', {}^{\bullet}E, \mathcal{A}]$, by subsumption with $H' \leqslant H$, and with and $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[H', {}^{\bullet}E, \mathcal{A}]$ derived by (PAR $\bullet$) from $\Gamma \vdash_\sigma P : \mathsf{Pro}[H', {}^{\bullet}E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H', {}^{\bullet}E, \mathcal{A}]$. From the last two judgments, by subsumption, one derives $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^{\bullet}E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^{\circ}E, \mathcal{A}]$, from which $\Gamma \vdash_\sigma P \mid Q : T$ derives by (PAR $\circ$).

The only other possibility is that $\Gamma \vdash_\sigma P \mid Q : T$ has been derived by (PAR $\circ$) from the judgements $\Gamma \vdash_\sigma P : \mathsf{Pro}[H', {}^{\bullet}E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H', {}^{\circ}E, \mathcal{A}]$, or from the judgements $\Gamma \vdash_\sigma P : \mathsf{Pro}[H', {}^{\circ}E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H', {}^{\bullet}E, \mathcal{A}]$ (with $H' \leqslant H$) followed by one or more subsumption steps. As in the previous cases, the proof follows by observing that the subsumption steps can be permuted up to the premises of the (PAR $\circ$) rule.

($\mathbf{c_3}$). $\Gamma \vdash_\sigma P \mid Q : T$ may have been derived from $\Gamma \vdash_\sigma P : \mathsf{Pro}[H', E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H', E, \mathcal{A}]$ by (PAR), for $H' \leqslant H$, followed by one or more subsumption steps. From these two judgments, by subsumption one derives $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, E, \mathcal{A}]$. Then the desired judgment derives by (PAR) followed by subsumption.

Otherwise, $\Gamma \vdash_\sigma P \mid Q : T$ must have been derived by (PAR $\mu \in \{\bullet, \circ, \vartriangle\}$) from the judgements $\Gamma \vdash_\sigma P : \mathsf{Pro}[H', {}^{\mu}E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H', {}^{\bullet}E, \mathcal{A}]$, or from the judgements $\Gamma \vdash_\sigma P : \mathsf{Pro}[H', {}^{\bullet}E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H', {}^{\mu}E, \mathcal{A}]$, with $H' \leqslant H$, followed by one or more subsumption steps. In all these cases, by subsumption, one derives $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^{\vartriangle}E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^{\bullet}E, \mathcal{A}]$, or $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^{\bullet}E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^{\vartriangle}E, \mathcal{A}]$. The the judgement $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[H, {}^{\vartriangle}E, \mathcal{A}]$ derives by (PAR $\vartriangle$).

Prefix:. The first part of the claim is obvious. For the second part, first observe that the rules for prefixes never derive types of the form $\mathsf{Pro}[H, {}^{\vartriangle}E, \mathcal{A}]$. Then, the proof follows by an inspection of the typing rules for prefixes and by observing that any subsumption step based on a subtyping relation of the form $\mathsf{Pro}[H', E, \mathcal{A}] \leqslant \mathsf{Pro}[H, E, \mathcal{A}]$ permutes with each of the prefix rules.

Input/Output:. By an inspection of the typing rules. □

A second lemma proves a useful property of upward silent processes.

LEMMA UPWARD SILENT PROCESSES. If $\Gamma \vdash_\sigma P : \mathsf{Pro}[E, {}^{\vartriangle}H, \mathcal{A}]$ is derivable with $H = \mathsf{shh}$, then $\Gamma \vdash_\sigma P : \mathsf{Pro}[E, {}^{\bullet}H, \mathcal{A}]$.

PROOF. By induction on the derivation of $\Gamma \vdash_\sigma P : \mathsf{Pro}[E, {}^{\vartriangle}H, \mathcal{A}]$, and observing that P may not have either of the forms $(\tilde{x} : W)^{\uparrow}Q$ and $\langle M_1, \ldots, M_k\rangle^{\uparrow}Q$.

LEMMA in MOVES PRESERVE TYPES. If $\Gamma \vdash_\rho a[\mathsf{in}\ b.P \mid Q] \mid b[R] : T$ is derivable, then so is $\Gamma \vdash_\rho b[a[P \mid Q] \mid R] : T$.

PROOF. By Lemma B.1, the judgment in the hypothesis must have been derived from $\Gamma \vdash_\rho b[R] : \mathsf{Pro}[E, {}^{\bullet}F, \mathcal{A}]$ and $\Gamma \vdash_\rho a[\mathsf{in}\ b.P \mid Q] : \mathsf{Pro}[E, {}^{\bullet}F, \mathcal{A}]$, for suitable E and F exchanges and $\mathcal{A}$ access mode such that $\mathsf{Pro}[E, {}^{\bullet}F, \mathcal{A}] \leqslant T$. We first show that $\Gamma \vdash_\rho b[a[P \mid Q] \mid R] : \mathsf{Pro}[E, {}^{\bullet}F, \mathcal{A}]$ is derivable.

The proof is a case analysis of the possible types of the sub-terms of $a[\mathsf{in}\ b.P \mid Q]$ and $b[R]$, guided by Lemma B.1. From $\Gamma \vdash_\rho b[R] : \mathsf{Pro}[E, {}^{\bullet}F, \mathcal{A}]$, we know that $\Gamma \vdash b : \tau\mathsf{Amb}^{?}[I, L, \mathcal{B}]$ and $\Gamma \vdash_\tau R : \mathsf{Pro}[I, {}^{?}L, \mathcal{B}]$ for any exchange type I, and suitable τ, L and $\mathcal{B}$. Now we look at $a[\mathsf{in}\ b.P \mid Q]$ and the possible types of its components, as informed by Lemma B.1, and show that $\Gamma \vdash_\tau a[P \mid Q] : \mathsf{Pro}[I, {}^{\bullet}J, \mathcal{B}]$ for every J.

By Lemma B.1, we find $E' \leqslant E$ such that $\Gamma \vdash_\rho a[\mathsf{in}\ b.P \mid Q] : \mathsf{Pro}[E', {}^{\bullet}F, \mathcal{A}]$ is derivable by any of the sets of assumptions defined by cases $(\mathbf{a_1})$ – $(\mathbf{a_3})$. We next consider those cases.

$(\mathbf{a_1})$ In this case $\Gamma \vdash a : \sigma\mathsf{Amb}[H, E', \mathcal{C}]$, and $\Gamma \vdash_\sigma \mathsf{in}\ b.P \mid Q : \mathsf{Pro}[H, E', \mathcal{C}]$ (in fact, we also have $\mathcal{P}(\sigma, \rho, \mathcal{C})$, but we may disregard this hypothesis, as it follows by the proof below). Let $T = \mathsf{Pro}[H, E', \mathcal{C}]$: by $(\mathbf{c_1})$ we know that $\Gamma \vdash_\sigma \mathsf{in}\ b.P : T$ and $\Gamma \vdash_\sigma Q : T$, and by $(\mathbf{p_1})$ that $\Gamma \vdash \mathsf{in}\ b : \sigma\mathsf{Cap}[E', \mathcal{C}]$ and $\Gamma \vdash_\sigma P : T$. From the former judgment and from $\Gamma \vdash b : \tau\mathsf{Amb}^{?}[I, J, \mathcal{B}]$, an inspection of the rule (IN) shows that $E' \leqslant I$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$. From $\Gamma \vdash_\sigma P : T$ and from $\Gamma \vdash_\sigma Q : T$, one has $\Gamma \vdash_\sigma P \mid Q : T$ by (PAR). From the last judgment and from $\Gamma \vdash a : \sigma\mathsf{Amb}[H, E', \mathcal{C}]$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$, by (AMB), one derives $\Gamma \vdash_\tau a[P \mid Q] : \mathsf{Pro}[E', {}^{\bullet}J, \mathcal{B}]$. Then $\Gamma \vdash_\tau a[P \mid Q] : \mathsf{Pro}[I, {}^{\bullet}J, \mathcal{B}]$ by subsumption, as desired.

$(\mathbf{a_2})$ In this case $\Gamma \vdash a : \sigma\mathsf{Amb}^{\circ}[H, E', \mathcal{C}]$ and $\Gamma \vdash_\sigma \mathsf{in}\ b.P \mid Q : \mathsf{Pro}[H, {}^{\vartriangle}E', \mathcal{C}]$. By Lemma B.1 we can assume that $\Gamma \vdash_\sigma \mathsf{in}\ b.P \mid Q : \mathsf{Pro}[H, {}^{\vartriangle}E', \mathcal{C}]$ derives from the three sets of assumptions defined by case $(\mathbf{c_3})$, which we consider below.

$(\mathbf{c_{3.1}})$ $\Gamma \vdash_\sigma \mathsf{in}\ b.P : \mathsf{Pro}[H, {}^{\bullet}E', \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^{\vartriangle}E', \mathcal{C}]$. From $\Gamma \vdash_\sigma \mathsf{in}\ b.P : \mathsf{Pro}[H, {}^{\bullet}E', \mathcal{C}]$, by $(\mathbf{p_1})$ we have $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^{\bullet}E', \mathcal{C}]$ and $\Gamma \vdash \mathsf{in}\ b : \sigma\mathsf{Cap}[E', \mathcal{C}]$. From the last judgment, and the typing of b an inspection of the rule (IN) shows that $E' \leqslant I$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$. From $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^{\bullet}E', \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^{\vartriangle}E', \mathcal{C}]$ one derives $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[H, {}^{\vartriangle}E', \mathcal{C}]$ by (PAR $\vartriangle$). From the last judgment and from $\Gamma \vdash a : \sigma\mathsf{Amb}^{\circ}[H, E', \mathcal{C}]$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$, one derives $\Gamma \vdash_\tau a[P \mid Q] : \mathsf{Pro}[E', {}^{\bullet}J, \mathcal{B}]$ by (AMB ${}^{\vartriangle}$). Now, $\Gamma \vdash_\sigma a[P \mid Q] : \mathsf{Pro}[I, {}^{\bullet}J, \mathcal{B}]$ derives by subsumption.

($\mathbf{c_{3.2}}$) $\Gamma \vdash_\sigma \mathsf{in}\ b.P : \mathsf{Pro}[H, {}^{\triangle}E', \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^{\bullet}E', \mathcal{C}]$. By Lemma B.1, we may assume that $\Gamma \vdash_\sigma \mathsf{in}\ b.P : \mathsf{Pro}[H, {}^{\triangle}E', \mathcal{C}]$ has been derived from the two sets of assumptions defined by case ($\mathbf{p_2}$). First observe that for all exchanges F and access modes $\mathcal{C}$, $\Gamma \vdash^{\circ} \mathsf{in}\ b : \sigma\mathsf{Cap}[F, \mathcal{C}]$ implies $\Gamma \vdash \mathsf{in}\ b : \sigma\mathsf{Cap}[F, \mathcal{C}]$. Then we can reason as follows.

In ($\mathbf{p_{2.1}}$), one has $\Gamma \vdash \mathsf{in}\ b : \overline{\sigma}\mathsf{Cap}[G, \mathcal{D}]$ and $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^{\circ}E', \mathcal{C}]$, with no constraint on the relationship between the exchanges G and E', the access modes $\mathcal{C}$ and $\mathcal{D}$, and the security levels $\overline{\sigma}$ and σ. From $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^{\circ}E', \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^{\bullet}E', \mathcal{C}]$, one has $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[H, {}^{\circ}E', \mathcal{C}]$ by (PAR $\circ$). Then, $\Gamma \vdash_\tau a[P \mid Q] : \mathsf{Pro}[I, {}^{\bullet}J, \mathcal{B}]$ derives directly by (AMB $\circ$) from $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[H, {}^{\circ}E', \mathcal{C}]$ and $\Gamma \vdash a : \sigma\mathsf{Amb}^{\circ}[H, E', \mathcal{C}]$.

In ($\mathbf{p_{2.2}}$) one has $\Gamma \vdash \mathsf{in}\ b : \sigma\mathsf{Cap}[E', \mathcal{C}]$ and $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^{\triangle}E', \mathcal{C}]$. From the former judgment, an inspection of the rule (IN) shows that $E' \leqslant I$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$. From $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^{\triangle}E', \mathcal{C}]$ and from $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^{\bullet}E', \mathcal{C}]$, one derives $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[H, {}^{\triangle}E', \mathcal{C}]$ by (PAR $\triangle$), and then $\Gamma \vdash_\tau a[P \mid Q] : \mathsf{Pro}[I, {}^{\bullet}J, \mathcal{B}]$ by (AMB $^{\triangle}$), which is applicable since $\mathcal{P}(\sigma, \tau, \mathcal{C})$, followed by subsumption.

($\mathbf{c_{3.3}}$) $\Gamma \vdash_\sigma \mathsf{in}\ b.P : \mathsf{Pro}[H, E', \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, E', \mathcal{C}]$. This case is similar to the case ($\mathbf{a_1}$) proved above, with the difference that now a is a pilot ambient. Reasoning as in that case, one derives $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[H, E', \mathcal{C}]$, and then $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[H, {}^{\triangle}E', \mathcal{C}]$ by subsumption, for $E' \leqslant I$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$. Now, from the last judgment and from $\Gamma \vdash a : \sigma\mathsf{Amb}^{\circ}[H, E', \mathcal{C}]$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$ we derive $\Gamma \vdash_\tau a[P \mid Q] : \mathsf{Pro}[E', {}^{\bullet}J, \mathcal{B}]$ by (AMB $\triangle$) and then, by subsumption $\Gamma \vdash_\tau a[P \mid Q] : \mathsf{Pro}[I, {}^{\bullet}J, \mathcal{B}]$.

($\mathbf{a_3}$) In this case $\Gamma \vdash a : \sigma\mathsf{Amb}^{\circ}[H, K, \mathcal{C}]$ and $\Gamma \vdash_\sigma \mathsf{in}\ b.P \mid Q : \mathsf{Pro}[H, {}^{\circ}K, \mathcal{C}]$. By Lemma B.1 we can assume that $\Gamma \vdash_\sigma \mathsf{in}\ b.P \mid Q : \mathsf{Pro}[H, {}^{\circ}K, \mathcal{C}]$ has been derived from the two pairs of assumptions defined by case ($\mathbf{c_2}$), which we consider below.

($\mathbf{c_{2.1}}$) $\Gamma \vdash_\sigma \mathsf{in}\ b.P : \mathsf{Pro}[H, {}^{\bullet}K, \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^{\circ}K, \mathcal{C}]$. By ($\mathbf{p_1}$) we know that $\Gamma \vdash \mathsf{in}\ b : \sigma\mathsf{Cap}[K, \mathcal{C}]$ and $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^{\bullet}K, \mathcal{C}]$. Now, from $\Gamma \vdash_\sigma P : \mathsf{Pro}[H, {}^{\bullet}K, \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^{\circ}K, \mathcal{C}]$ one derives $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[H, {}^{\circ}K, \mathcal{C}]$ by (PAR $\circ$). Now, from the last judgment and $\Gamma \vdash a : \mathsf{Amb}^{\circ}[H, K, \mathcal{C}]$, one derives $\Gamma \vdash_\tau a[P \mid Q] : \mathsf{Pro}[I, {}^{\bullet}J, \mathcal{B}]$ directly by (AMB $\circ$).

($\mathbf{c_{2.2}}$) $\Gamma \vdash_\sigma \mathsf{in}\ b.P : \mathsf{Pro}[H, {}^{\circ}K, \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[H, {}^{\bullet}K, \mathcal{C}]$. The proof further splits in the two subcases defined by ($\mathbf{p_2}$) and proceeds as in case ($\mathbf{c_{3.2}}$) above, with E' replaced by K.

From the previous analysis we have $\Gamma \vdash_\tau a[P \mid Q] : \mathsf{Pro}[I, {}^{\bullet}J, \mathcal{B}]$ for any J. From the hypothesis, we had inferred that $\Gamma \vdash_\tau R : \mathsf{Pro}[I, {}^{?}L, \mathcal{B}]$, for a suitable L. Choosing $J = L$, by the appropriate (PAR ?) rule, we then derive $\Gamma \vdash_\tau R \mid a[P \mid Q] : \mathsf{Pro}[I, {}^{?}L, \mathcal{B}]$. From the last judgment and from $\Gamma \vdash b : \tau\mathsf{Amb}^{?}[I, L, \mathcal{B}]$ we conclude $\Gamma \vdash_\rho b[R \mid a[P \mid Q]] : \mathsf{Pro}[E, {}^{\bullet}F, \mathcal{A}]$ using the appropriate (AMB ?) rule (the same rule used in the derivation of $\Gamma \vdash_\rho b[R] : \mathsf{Pro}[E, {}^{\bullet}F, \mathcal{A}]$). ☐

LEMMA out MOVES PRESERVE TYPES. If $\Gamma \vdash_\rho a[b[\mathsf{out}\ a.P \mid Q] \mid R] : T$ is derivable, then so is $\Gamma \vdash_\rho b[P \mid Q] \mid a[R] : T$.

PROOF. By Lemma B.1, there exist E, F exchanges, and $\mathcal{B}$ access mode such that $\mathsf{Pro}[E, {}^{\bullet}F, \mathcal{B}] \leqslant T$ and the judgment in the hypothesis must have been derived from $\Gamma \vdash_\rho a[b[\mathsf{out}\ a.P \mid Q] \mid R] : \mathsf{Pro}[E, {}^{\bullet}F, \mathcal{B}]$. This implies that $\Gamma \vdash_\rho a[R] : \mathsf{Pro}[E, {}^{\bullet}F, \mathcal{B}]$ is also derivable. To prove the lemma, we thus need to show that $\Gamma \vdash_\rho b[P \mid Q] : \mathsf{Pro}[E, {}^{\bullet}F, \mathcal{B}]$.

We distinguish two cases, depending on the type of a, namely: $\Gamma \vdash a : \tau\mathsf{Amb}^{?}[I,E,\mathcal{C}]$, or $\Gamma \vdash a : \tau\mathsf{Amb}^{\circ}[I,K,\mathcal{C}]$ for some exchanges I and K, security level τ and access mode $\mathcal{C}$ (note that Lemma B.1 ensures that E can be chosen so that a has the indicated types). For each of the two cases, we look at $b[\mathsf{out}\ a.P \mid Q] \mid R$ and at the possible types of its components, as informed by Lemma B.1.

Case $\Gamma \vdash a : \tau\mathsf{Amb}^{?}[I,E,\mathcal{C}]$. From $\Gamma \vdash_{\rho} a[b[\mathsf{out}\ a.P \mid Q] \mid R] : \mathsf{Pro}[E, {}^{\bullet}F, \mathcal{A}]$, by Lemma B.1 ($(\mathbf{a_1})$ and $(\mathbf{a_2})$) it follows that $\mathcal{P}(\tau,\rho,\mathcal{C})$ and $\Gamma \vdash_{\tau} b[\mathsf{out}\ a.P \mid Q] \mid R : \mathsf{Pro}[I, {}^{\mu}E, \mathcal{C}]$ for any I, and μ either absent or equal to $\vartriangle$.

Now, by two applications of Lemma B.1 (to the parallel composition, and then to the process in ambient form), it follows that there exists $H \leqslant I$ such that $\Gamma \vdash_{\tau} b[\mathsf{out}\ a.P \mid Q] : \mathsf{Pro}[H, {}^{\bullet}E, \mathcal{C}]$ is derivable by any of the three sets of assumptions defined by cases $(\mathbf{a_1})$ – $(\mathbf{a_3})$. We consider those cases below.

$(\mathbf{a_1})$ In this case $\Gamma \vdash b : \sigma\mathsf{Amb}[L,H,\mathcal{A}]$, and $\Gamma \vdash_{\sigma} \mathsf{out}\ a.P \mid Q : T$ with $T = \mathsf{Pro}[L,H,\mathcal{A}]$ and $\mathcal{P}(\sigma,\tau,\mathcal{A})$. By $(\mathbf{c_1})$ we know that $\Gamma \vdash_{\sigma} \mathsf{out}\ a.P : T$ and $\Gamma \vdash_{\sigma} Q : T$. From the first judgment, by $(\mathbf{p_1})$, one has $\Gamma \vdash \mathsf{out}\ a : \sigma\mathsf{Cap}[H,\mathcal{A}]$ and $\Gamma \vdash_{\sigma} P : T$. An inspection of the typing rules shows that $\Gamma \vdash \mathsf{out}\ a : \sigma\mathsf{Cap}[H,\mathcal{A}]$ must have been derived by the rule (OUT). Then the typing of a is, in fact, $\Gamma \vdash a : \tau\mathsf{Amb}[H,E,\mathcal{C}]$, and furthermore $H \leqslant E$ and $\mathcal{A} \leqslant \mathcal{C}$. Since $\mathcal{P}$ is stable, by assumption, from $\mathcal{P}(\tau,\rho,\mathcal{C})$ and $\mathcal{A} \leqslant \mathcal{C}$ we have $\mathcal{P}(\tau,\rho,\mathcal{A})$: this, together with $\mathcal{P}(\sigma,\tau,\mathcal{A})$, implies $\mathcal{P}(\sigma,\rho,\mathcal{A})$. Now, from $\Gamma \vdash_{\sigma} P : T$ and $\Gamma \vdash_{\sigma} Q : T$ one has $\Gamma \vdash_{\sigma} P \mid Q : T$ by (PAR), and from this judgment and from $\Gamma \vdash b : \sigma\mathsf{Amb}[L,H,\mathcal{A}]$ one derives $\Gamma \vdash_{\rho} b[P \mid Q] : \mathsf{Pro}[H, {}^{\bullet}F, \mathcal{B}]$ by (AMB).
Then $\Gamma \vdash_{\rho} b[P \mid Q] : \mathsf{Pro}[E, {}^{\bullet}F, \mathcal{B}]$ derives by subsumption, given that $H \leqslant E$.

$(\mathbf{a_2})$ In this case $\Gamma \vdash b : \sigma\mathsf{Amb}^{\circ}[L,H,\mathcal{A}]$ and $\Gamma \vdash_{\sigma} \mathsf{out}\ a.P \mid Q : \mathsf{Pro}[L, {}^{\vartriangle}H, \mathcal{A}]$, with $\mathcal{P}(\sigma,\tau,\mathcal{A})$. By Lemma B.1 we can assume that $\Gamma \vdash_{\sigma} \mathsf{out}\ a.P \mid Q : \mathsf{Pro}[L, {}^{\vartriangle}H, \mathcal{A}]$ derives from one of the three sets of assumptions defined by case $(\mathbf{c_3})$. We consider these cases below.

$(\mathbf{c_{3.1}})$ $\Gamma \vdash_{\sigma} \mathsf{out}\ a.P : \mathsf{Pro}[L, {}^{\bullet}H, \mathcal{A}]$ and $\Gamma \vdash_{\sigma} Q : \mathsf{Pro}[L, {}^{\vartriangle}H, \mathcal{A}]$. From $\Gamma \vdash_{\sigma} \mathsf{out}\ a.P : \mathsf{Pro}[L, {}^{\bullet}H, \mathcal{A}]$, by $(\mathbf{p_1})$ we have $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[L, {}^{\bullet}H, \mathcal{A}]$ and $\Gamma \vdash \mathsf{out}\ a : \sigma\mathsf{Cap}[H,\mathcal{A}]$, with the last judgment derived by the rule (OUT). Thus the typing of a must be $\Gamma \vdash a : \tau\mathsf{Amb}[I,E,\mathcal{C}]$, and moreover $H \leqslant E$ and $\mathcal{A} \leqslant \mathcal{C}$. Reasoning as in case $(\mathbf{a_1})$ above, it follows that $\mathcal{P}(\sigma,\rho,\mathcal{A})$. From $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[L, {}^{\bullet}H, \mathcal{A}]$ and $\Gamma \vdash_{\sigma} Q : \mathsf{Pro}[L, {}^{\vartriangle}H, \mathcal{A}]$ one derives $\Gamma \vdash_{\sigma} P \mid Q : \mathsf{Pro}[L, {}^{\vartriangle}H, \mathcal{A}]$ by (PAR $\vartriangle$). From the last judgment and from $\Gamma \vdash b : \sigma\mathsf{Amb}^{\circ}[L,H,\mathcal{A}]$, one derives by (AMB $^{\vartriangle}$) $\Gamma \vdash_{\rho} b[P \mid Q] : \mathsf{Pro}[H, {}^{\bullet}J, \mathcal{B}]$ for any J and thus, in particular, for $J = F$. Now, $\Gamma \vdash_{\rho} b[P \mid Q] : \mathsf{Pro}[E, {}^{\bullet}F, \mathcal{B}]$ derives by subsumption, as $H \leqslant E$.

$(\mathbf{c_{3.2}})$ $\Gamma \vdash_{\sigma} \mathsf{out}\ a.P : \mathsf{Pro}[L, {}^{\vartriangle}H, \mathcal{A}]$ and $\Gamma \vdash_{\sigma} Q : \mathsf{Pro}[L, {}^{\bullet}H, \mathcal{A}]$. By Lemma B.1, we may assume that $\Gamma \vdash_{\sigma} \mathsf{out}\ a.P : \mathsf{Pro}[L, {}^{\vartriangle}H, \mathcal{A}]$ has been derived from the two pairs of assumptions defined by case $(\mathbf{p_2})$.

In $(\mathbf{p_{2.1}})$, one has $\Gamma \vdash^{\circ} \mathsf{out}\ a : \overline{\sigma}\mathsf{Cap}[G,\mathcal{D}]$ and $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[L, {}^{\circ}H, \mathcal{A}]$, with G any exchange, $\mathcal{D}$ any access mode, and $\overline{\sigma}$ any security level. From $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[L, {}^{\circ}H, \mathcal{A}]$ and From $\Gamma \vdash_{\sigma} Q : \mathsf{Pro}[L, {}^{\bullet}H, \mathcal{A}]$ one derives $\Gamma \vdash_{\sigma} P \mid Q : \mathsf{Pro}[L, {}^{\circ}H, \mathcal{A}]$ by (PAR $\circ$). Then, we derive $\Gamma \vdash_{\rho} b[P \mid Q] : \mathsf{Pro}[E, {}^{\bullet}F, \mathcal{B}]$ directly by (AMB $\circ$).

In $(\mathbf{p_{2.2}})$, one has $\Gamma \vdash^{\circ} \mathsf{out}\ a : \sigma\mathsf{Cap}[H,\mathcal{A}]$ and $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[L, {}^{\vartriangle}H, \mathcal{A}]$.
If the typing of a is $\Gamma \vdash a : \tau\mathsf{Amb}[I,E,\mathcal{C}]$, then $\Gamma \vdash^{\circ} \mathsf{out}\ a : \sigma\mathsf{Cap}[H,\mathcal{A}]$ must have been derived from $\Gamma \vdash \mathsf{out}\ a : \sigma\mathsf{Cap}[H,\mathcal{A}]$, which implies that $H \leqslant E$ and $\mathcal{A} \leqslant \mathcal{C}$. Reasoning

as in case ($\mathbf{a_1}$), it follows again that $\mathcal{P}(\sigma,\rho,\mathcal{A})$. Then, from $\Gamma \vdash_\sigma P : \mathsf{Pro}[L, {}^\vartriangle H, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[L, {}^\bullet H, \mathcal{A}]$ one derives $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[L, {}^\vartriangle H, \mathcal{A}]$ by (PAR $\vartriangle$). From this last judgment and from $\Gamma \vdash b : \sigma\mathsf{Amb}^\circ[L,H,\mathcal{A}]$, one derives $\Gamma \vdash_\rho b[P \mid Q] : [E, {}^\bullet F, \mathcal{B}]$ by (AMB $\vartriangle$) and subsumption with $H \leqslant E$.
Instead, if the typing of a is $\Gamma \vdash a : \sigma\mathsf{Amb}^\circ[H,E,\mathcal{A}]$ then $\Gamma \vdash^\circ \mathsf{out}\ a : \sigma\mathsf{Cap}[H,\mathcal{A}]$ implies $H = \mathsf{shh}$. But then, from the hypothesis $\Gamma \vdash_\sigma P : \mathsf{Pro}[L, {}^\vartriangle H, \mathcal{A}]$, by Lemma B.2, it follows that also $\Gamma \vdash_\sigma P : \mathsf{Pro}[L, {}^\circ H, \mathcal{A}]$ is derivable. Then $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[L, {}^\circ H, \mathcal{A}]$ is derivable by (PAR $\circ$). Now $\Gamma \vdash_\rho b[P \mid Q] : \mathsf{Pro}[H, {}^\bullet F, \mathcal{B}]$ derives by (AMB $\circ$), and $\Gamma \vdash_\rho b[P \mid Q] : \mathsf{Pro}[E, {}^\bullet F, \mathcal{B}]$ by subsumption.

($\mathbf{c_{3.3}}$) $\Gamma \vdash_\sigma \mathsf{out}\ a.P : \mathsf{Pro}[L,H,\mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[L,H,\mathcal{A}]$. This case has the same hypothesis as the case ($\mathbf{a_1}$) above, save that b is typed as a pilot ambient. Reasoning as in that case, one derives $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[L,H,\mathcal{A}]$, and then $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[L, {}^\vartriangle H, \mathcal{A}]$ by subsumption, with $H \leqslant E$. The proof proceeds as in ($\mathbf{a_1}$): only, it uses (AMB $\vartriangle$) instead of (AMB).

($\mathbf{a_3}$) In this case $\Gamma \vdash b : \sigma\mathsf{Amb}^\circ[L,K,\mathcal{A}]$ and $\Gamma \vdash_\sigma \mathsf{out}\ a.P \mid Q : \mathsf{Pro}[L, {}^\circ K, \mathcal{A}]$. By Lemma B.1 we can assume that $\Gamma \vdash_\sigma \mathsf{out}\ a.P \mid Q : \mathsf{Pro}[L, {}^\circ K, \mathcal{A}]$ has been derived from any of the two sets of assumptions defined by case ($\mathbf{c_2}$), which we consider below.

($\mathbf{c_{2.1}}$) $\Gamma \vdash_\sigma \mathsf{out}\ a.P : \mathsf{Pro}[L, {}^\bullet K, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[L, {}^\circ K, \mathcal{A}]$. By ($\mathbf{p_1}$) we know that $\Gamma \vdash_\sigma P : \mathsf{Pro}[L, {}^\bullet K, \mathcal{A}]$. From $\Gamma \vdash_\sigma P : \mathsf{Pro}[L, {}^\bullet K, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[L, {}^\circ K, \mathcal{A}]$ one derives $\Gamma \vdash_\sigma P \mid Q : \mathsf{Pro}[L, {}^\circ K, \mathcal{A}]$ by (PAR $\circ$). From the last judgment and from $\Gamma \vdash b : \sigma\mathsf{Amb}^\circ[L,K,\mathcal{A}]$, one derives $\Gamma \vdash_\rho b[P \mid Q] : \mathsf{Pro}[E, {}^\bullet F, \mathcal{B}]$ directly by (AMB $\circ$).

($\mathbf{c_{2.2}}$) $\Gamma \vdash_\sigma \mathsf{out}\ a.P : \mathsf{Pro}[L, {}^\circ K, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[L, {}^\bullet K, \mathcal{A}]$. The proof further splits in the two subcases defined by ($\mathbf{p_2}$) and proceeds as in case ($\mathbf{c_{3.2}}$) above, with H replaced by K.

Case $\Gamma \vdash a : \tau\mathsf{Amb}^\circ[I,K,\mathcal{A}]$. From $\Gamma \vdash_\rho a[b[\mathsf{out}\ a.P \mid Q] \mid R] : \mathsf{Pro}[E, {}^\bullet F, \mathcal{A}]$, by Lemma B.1 ($\mathbf{a_1}$) and ($\mathbf{a_2}$), and then ($\mathbf{c_1}$) – ($\mathbf{c_3}$), it follows that there exists $H \leqslant I$ such that $\Gamma \vdash_\tau b[\mathsf{out}\ a.P \mid Q] : \mathsf{Pro}[H, {}^\bullet K, \mathcal{A}]$ is derivable by any of the three sets of assumptions defined by cases ($\mathbf{a_1}$) – ($\mathbf{a_3}$). We consider those cases below: as we shall see, most of them are vacuous, given the typing of a as a pilot ambient.

($\mathbf{a_1}$) In this case $\Gamma \vdash b : \sigma\mathsf{Amb}[L,H,\mathcal{A}]$ and $\Gamma \vdash_\sigma \mathsf{out}\ a.P \mid Q : \mathsf{Pro}[L,H,\mathcal{A}]$. This is one of the vacuous cases. To see that, note that $\Gamma \vdash_\sigma \mathsf{out}\ a.P \mid Q : \mathsf{Pro}[L,H,\mathcal{A}]$ implies, by ($\mathbf{c_1}$) and ($\mathbf{p_1}$), that $\Gamma \vdash \mathsf{out}\ a : \sigma\mathsf{Cap}[H,\mathcal{A}]$ is derivable. An inspection of the typing rules shows that this is not possible, as we are currently assuming that $\Gamma \vdash a : \sigma\mathsf{Amb}^\circ[H,K,\mathcal{A}]$, and hence the only derivable judgments for $\mathsf{out}\ a$ are of the form $\Gamma \vdash^\circ \mathsf{out}\ a : \sigma\mathsf{Cap}[\mathsf{shh},\mathcal{A}]$ for some mode $\mathcal{A}$.

($\mathbf{a_2}$) In this case $\Gamma \vdash b : \sigma\mathsf{Amb}^\circ[L,H,\mathcal{A}]$ and $\Gamma \vdash_\sigma \mathsf{out}\ a.P \mid Q : \mathsf{Pro}[L, {}^\vartriangle H, \mathcal{A}]$. By Lemma B.1 we can assume that $\Gamma \vdash_\sigma \mathsf{out}\ a.P \mid Q : \mathsf{Pro}[L, {}^\vartriangle H, \mathcal{A}]$ derives from any of the three sets of assumptions defined by case ($\mathbf{c_3}$). We consider these cases below.
In case ($\mathbf{c_{3.1}}$) one has $\Gamma \vdash_\sigma \mathsf{out}\ a.P : \mathsf{Pro}[L, {}^\bullet H, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[L, {}^\vartriangle H, \mathcal{A}]$: this is another vacuous case, for the reason given above. Similarly, case ($\mathbf{c_{3.3}}$) is vacuous as $\Gamma \vdash_\sigma \mathsf{out}\ a.P : \mathsf{Pro}[L,H,\mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[L,H,\mathcal{A}]$.
In case ($\mathbf{c_{3.2}}$), $\Gamma \vdash_\sigma \mathsf{out}\ a.P : \mathsf{Pro}[L, {}^\vartriangle H, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : \mathsf{Pro}[L, {}^\bullet H, \mathcal{A}]$. By Lemma B.1, we may assume that $\Gamma \vdash_\sigma \mathsf{out}\ a.P : \mathsf{Pro}[L, {}^\vartriangle H, \mathcal{A}]$ has been derived from one of the two pairs of assumptions defined by case ($\mathbf{p_2}$).

In $(\mathbf{p_{2.1}})$, one has $\Gamma \vdash^{\circ} \mathsf{out}\ a : \overline{\sigma}\mathsf{Cap}[G,\mathcal{D}]$ and $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[L,{}^{\circ}H,\mathcal{A}]$, for any G, $\mathcal{D}$, and $\overline{\sigma}$. From $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[L,{}^{\circ}H,\mathcal{A}]$ and from $\Gamma \vdash_{\sigma} Q : \mathsf{Pro}[L,{}^{\bullet}H,\mathcal{A}]$ one derives $\Gamma \vdash_{\sigma} P \mid Q : \mathsf{Pro}[L,{}^{\circ}H,\mathcal{A}]$ by (PAR $\circ$). Then, $\Gamma \vdash_{\rho} b[P \mid Q] : \mathsf{Pro}[E,{}^{\bullet}F,\mathcal{B}]$ derives directly by (AMB $\circ$).

In $(\mathbf{p_{2.2}})$, one has $\Gamma \vdash^{\circ} \mathsf{out}\ a : \sigma\mathsf{Cap}[H,\mathcal{A}]$ and $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[L,{}^{\vartriangle}H,\mathcal{A}]$. The typing of a and $\Gamma \vdash^{\circ} \mathsf{out}\ a : \sigma\mathsf{Cap}[H,\mathcal{A}]$ imply that $H = \mathsf{shh}$. But then, from the hypothesis $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[L,{}^{\vartriangle}H,\mathcal{A}]$, by Lemma B.2, it follows that also $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[L,{}^{\circ}H,\mathcal{A}]$ is derivable. Now, $\Gamma \vdash_{\sigma} P \mid Q : \mathsf{Pro}[L,{}^{\circ}H,\mathcal{A}]$ is derivable by (PAR $\circ$), and then $\Gamma \vdash_{\rho} b[P \mid Q] : \mathsf{Pro}[E,{}^{\bullet}F,\mathcal{B}]$ directly by (AMB $\circ$).

$(\mathbf{a_3})$ In this case $\Gamma \vdash b : \sigma\mathsf{Amb}^{\circ}[L,J,\mathcal{A}]$ and $\Gamma \vdash_{\sigma} \mathsf{out}\ a.P \mid Q : \mathsf{Pro}[L,{}^{\circ}J,\mathcal{A}]$. By Lemma B.1 we can assume that $\Gamma \vdash_{\sigma} \mathsf{out}\ a.P \mid Q : \mathsf{Pro}[L,{}^{\circ}J,\mathcal{A}]$. has been derived from one of the two pairs of assumptions defined by case $(\mathbf{c_2})$. Case $(\mathbf{c_{2.1}})$ is vacuous, as it implies $\Gamma \vdash_{\sigma} \mathsf{out}\ a.P : \mathsf{Pro}[L,{}^{\bullet}J,\mathcal{A}]$ and $\Gamma \vdash_{\sigma} Q : \mathsf{Pro}[L,{}^{\circ}J,\mathcal{A}]$.
In case $(\mathbf{c_{2.2}})$, $\Gamma \vdash_{\sigma} \mathsf{out}\ a.P : \mathsf{Pro}[L,{}^{\circ}J,\mathcal{A}]$ and $\Gamma \vdash_{\sigma} Q : \mathsf{Pro}[L,{}^{\bullet}J,\mathcal{A}]$. The proof further splits into the two subcases defined by $(\mathbf{p_2})$, namely: (i) $\Gamma \vdash^{\circ} \mathsf{out}\ a : \overline{\sigma}\mathsf{Cap}[G,\mathcal{D}]$ and $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[L,{}^{\circ}J,\mathcal{A}]$, and (ii) $\Gamma \vdash^{\circ} \mathsf{out}\ a : \sigma\mathsf{Cap}[J,\mathcal{A}]$ and $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[L,{}^{\vartriangle}J,\mathcal{A}]$. In the first subcase, the claim follows as in case $(\mathbf{p_{2.1}})$ above, by a final application of (AMB $\circ$). In the second, $J = \mathsf{shh}$ and the claim follows as in case $(\mathbf{p_{2.2}})$. $\square$

LEMMA TYPE ENVIRONMENTS. Let $\Gamma \vdash^{?} U : Z$ denote any of the judgments $\Gamma \vdash^{\circ} M : W$, $\Gamma \vdash M : W$ or $\Gamma \vdash_{\sigma} P : T$.

(1) If $\Gamma \vdash \diamond$ then $\Gamma' \vdash \diamond$ for every $\Gamma' \subseteq \Gamma$.
(2) If $\Gamma \vdash^{?} U : Z$ then $\Gamma \vdash \diamond$.
(3) If $\Gamma, x : W, \Gamma' \vdash^{?} U : Z$ and $x \notin fn(U)$ then $\Gamma, \Gamma' \vdash^{?} U : Z$.
(4) If $\Gamma, \vdash^{?} U : Z$ and $\Gamma, \Gamma' \vdash \diamond$ then $\Gamma, \Gamma' \vdash^{?} U : Z$.

PROOF. By induction on the derivation of the first judgments in each of the hypotheses.

LEMMA SUBSTITUTION.

(1) Assume $\Gamma, x_1{:}W_1, \ldots, x_k{:}W_k, \Gamma' \vdash^{?} M{:}W$. For all $N_1, \ldots N_k$, if $\forall i \in \{1 \ldots k\}$ $\Gamma, \Gamma' \vdash N_i{:}W_i$, then $\Gamma, \Gamma' \vdash^{?} M\{x_i := N_i\}{:}W$.
(2) Assume $\Gamma, x_1{:}W_1, \ldots, x_k{:}W_k, \Gamma' \vdash_{\sigma} P : T$. For all $N_1, \ldots N_k$, if $\forall i \in \{1 \ldots k\}$ $\Gamma, \Gamma' \vdash N_i : W_i$, then $\Gamma, \Gamma' \vdash_{\sigma} P\{x_i := N_i\} : T$.

PROOF. The proof is by induction on the derivations of first judgments in the hypotheses and a case analysis on the last applied rule. Most cases follow directly by the induction hypothesis: we give a proof of the representative cases.

(1) **(Projection)** The hypothesis is $\Gamma, x_1{:}W_1, \ldots, x_k{:}W_k, \Gamma' \vdash y : W$. We have two cases to consider. If $y = x_i$ for some $i \in \{1 \ldots k\}$, then it must be the case that $W = W_i$ and the claim follows from the hypothesis $\Gamma, \Gamma' \vdash N_i : W_i$. If $y \notin \{x_1, \ldots, x_k\}$, from $\Gamma, x_1{:}W_1, \ldots, x_k{:}W_k, \Gamma' \vdash y : W$, by Lemma B.5.3 we have $\Gamma, \Gamma' \vdash y : W$. This concludes the proof since since $y = y\{x_i := N_i\}$.

(In) The hypothesis is $\Gamma, x_1{:}W_1, \ldots, x_k{:}W_k, \Gamma' \vdash \mathsf{in}\ M : \sigma\mathsf{Cap}[G,\mathcal{B}]$, derived from the judgment $\Gamma, x_1{:}W_1, \ldots, x_k{:}W_k, \Gamma' \vdash M : \rho\mathsf{Amb}^{?}[F,E,\mathcal{A}]$ with $G \leqslant F$ and $\mathcal{P}(\sigma,\rho,\mathcal{B})$. By the induction hypothesis it follows that $\Gamma, \Gamma' \vdash M\{x_i := N_i\} : \rho\mathsf{Amb}^{?}[F,E,\mathcal{A}]$. Then, the desired judgment derives by an application of the rule (IN).

(2) **(Prefix)** The hypothesis is $\Gamma, x_1{:}W_1, \ldots, x_k{:}W_k, \Gamma' \vdash_\sigma M.P : \mathsf{Pro}[E, F, \mathcal{A}]$, derived from $\Gamma, x_1 : W_1, \ldots, x_k : W_k, \Gamma' \vdash M : \sigma\mathsf{Cap}[F, \mathcal{A}]$ and from $\Gamma, x_1 : W_1, \ldots, x_k : W_k, \Gamma' \vdash_\sigma P : \mathsf{Pro}[E, F, \mathcal{A}]$. By the induction hypothesis, $\Gamma, \Gamma' \vdash M\{x_i := N_i\} : \sigma\mathsf{Cap}[F, \mathcal{A}]$ and $\Gamma, \Gamma' \vdash_\sigma P\{x_i := N_i\} : \mathsf{Pro}[E, F, \mathcal{A}]$ are both derivable. Then, the judgement $\Gamma, \Gamma' \vdash_\sigma M\{x_i := N_i\}.P\{x_i := N_i\} : \mathsf{Pro}[E, F, \mathcal{A}]$ derives by (PREFIX).

(New) The hypothesis is $\Gamma, x_1{:}W_1, \ldots, x_k{:}W_k, \Gamma' \vdash_\sigma (\nu y{:}\rho\mathsf{Amb}^?[E, F, \mathcal{A}])P : T$, derived from $\Gamma, x_1{:}W_1, .., x_k{:}W_k, \Gamma', y{:}\rho\mathsf{Amb}^?[E, F, \mathcal{A}] \vdash_\sigma P : T$. By Lemma B.5.2, we have $\Gamma, x_1{:}W_1, .., x_k{:}W_k, \Gamma', y{:}\rho\mathsf{Amb}^?[E, F, \mathcal{A}] \vdash \diamond$, hence that $y \notin \{x_1, \ldots, x_k\}$. By the induction hypothesis we have that $\Gamma, \Gamma', y{:}\rho\mathsf{Amb}^?[E, F, \mathcal{A}] \vdash_\sigma P\{x_i := N_i\} : T$. Now, from the last judgment, by (NEW), one derives $\Gamma, \Gamma' \vdash_\sigma (\nu y{:}\rho\mathsf{Amb}^?[E, F, \mathcal{A}])(P\{x_i := N_i\}) : T$. This is the judgment we wished to derive, as $y \notin \{x_1, \ldots, x_n\}$ implies that $(\nu y{:}\rho\mathsf{Amb}^?[E, F, \mathcal{A}])(P\{x_i := N_i\}) = ((\nu y{:}\rho\mathsf{Amb}^?[E, F, \mathcal{A}])P)\{x_i := N_i\}$.

(Output N**)** The judgement in the hypothesis is $\Gamma, x_1{:}W_1, \ldots, x_k{:}W_k, \Gamma' \vdash_\sigma \langle M \rangle^N P : T$. By Lemma B.1.($\mathbf{o_2}$) there exist E, F, G, W, $\mathcal{A}$, and ρ such that the judgements $\Gamma, x_1{:}W_1, \ldots, x_k{:}W_k, \Gamma' \vdash N : \rho\mathsf{Amb}^?[W, G, \mathcal{A}]$, $\Gamma, x_1{:}W_1, \ldots, x_k{:}W_k, \Gamma' \vdash M : W$, and $\Gamma, x_1{:}W_1, \ldots, x_k{:}W_k, \Gamma' \vdash_\sigma P : \mathsf{Pro}[E, {}^\mu F, \mathcal{A}] \leqslant T$ are derivable and $\mathcal{P}(\sigma, \rho, \mathsf{w})$ holds true. By the induction hypothesis $\Gamma, \Gamma' \vdash N\{x_i := N_i\} : \rho\mathsf{Amb}^?[W, G, \mathcal{A}]$, and $\Gamma, \Gamma' \vdash M\{x_i := N_i\} : W$ and $\Gamma, \Gamma' \vdash_\sigma P\{x_i := N_i\} : \mathsf{Pro}[E, {}^\mu F, \mathcal{A}]$ are all derivable. Then, by (OUTPUT N) and subsumption one derives the desired judgment $\Gamma, \Gamma' \langle M\{x_i := N_i\} \rangle^{N\{x_i := N_i\}} P\{x_i := N_i\} : T$.

LEMMA SYNCHRONOUS EXCHANGE PRESERVES TYPES.

(1) If $\Gamma \vdash_\sigma (x_1 : W_1, \ldots, x_k : W_k)P \mid \langle M_1, \ldots, M_k \rangle Q : T$, then $\Gamma \vdash_\sigma P\{x_i := M_i\} \mid Q : T$

(2) If $\Gamma \vdash_\sigma (x_1 : W_1, \ldots, x_k : W_k)^n P \mid n[\langle M_1, \ldots, M_k \rangle Q \mid R] : T$,
then $\Gamma \vdash_\sigma P\{x_i := M_i\} \mid n[Q \mid R] : T$

(3) If $\Gamma \vdash_\sigma \langle M_1, \ldots, M_k \rangle P \mid n[(x_1 : W_1, \ldots, x_k : W_k)^\uparrow Q \mid R] : T$,
then $\Gamma \vdash_\sigma P \mid n[Q\{x_i := M_i\} \mid R] : T$

(4) If $\Gamma \vdash_\sigma \langle M_1, \ldots, M_k \rangle^n P \mid n[(x_1 : W_1, \ldots, x_k : W_k)Q \mid R] : T$,
then $\Gamma \vdash_\sigma P \mid n[Q\{x_i := M_i\} \mid R] : T$

(5) If $\Gamma \vdash_\sigma (x_1 : W_1, \ldots, x_k : W_k)P \mid n[\langle M_1, \ldots, M_k \rangle^\uparrow Q \mid R] : T$,
then $\Gamma \vdash_\sigma P\{x_i := M_i\} \mid n[Q \mid R] : T$

PROOF. We only give three cases as representative: 1 (local exchange), 2 (downward input) and 5 (upward output). The remaining cases are handled similarly.

1.. By repeated applications of Lemma B.1 (on the parallel composition, and then on the component processes), it follows that $T = \mathsf{Pro}[W_1 \times \cdots \times W_k, {}^? F, \mathcal{A}]$, for some access $\mathcal{A}$, and exchange type ${}^?F$. The judgment in the hypothesis must have been derived from $\Gamma, x_1 : W_1, \ldots x_k : W_k \vdash_\sigma P : \mathsf{Pro}[W_1 \times \cdots \times W_k, {}^{?_1} F, \mathcal{A}]$, from $\Gamma \vdash_\sigma M_i : W_i$ $(i = 1..k)$ and from $\Gamma \vdash_\sigma Q : \mathsf{Pro}[W_1 \times \cdots \times W_k, {}^{?_2} F, \mathcal{A}]$ by a rule (PAR $?_h$) where h is either 1 or 2. From the first two judgments, by Lemma B.6, we know that $\Gamma \vdash_\sigma P\{x_i := M_i\} : \mathsf{Pro}[W_1 \times \cdots \times W_k, {}^{?_1} F, \mathcal{A}]$. Then $\Gamma \vdash_\sigma P\{x_i := M_i\} \mid Q : \mathsf{Pro}[W_1 \times \cdots \times W_k, {}^? F, \mathcal{A}]$ derives directly by the given (PAR $?_h$) rule.

2.. The judgment in the hypothesis must have been derived from the judgements $\Gamma \vdash_\sigma (x_1 : W_1, \ldots, x_k : W_k)^n P : T'$ and $\Gamma \vdash_\sigma n[\langle M_1, \ldots, M_k \rangle Q \mid R] : T''$ for appropriate T' and

T''. Let Ξ be the (partial) derivation from these two judgments to the judgment in the hypothesis.

From $\Gamma \vdash_\sigma (x_1 : W_1,\dots,x_k : W_k)^n P : T'$, by Lemma B.1($\mathbf{i_2}$) and subsumption, we know that $\Gamma, x_1 : W_1,\dots,x_k : W_k \vdash_\sigma P : T'$, and $\Gamma \vdash n : \rho\mathsf{Amb}^?[W_1 \times \cdots \times W_k, E, \mathcal{A}]$ for some E, ρ and $\mathcal{A}$ (we also know that $\mathcal{P}(\sigma,\rho,\mathsf{r})$, but this is only useful in the proof of type soundness, not here). We take the case when $\Gamma \vdash n : \rho\mathsf{Amb}[W_1 \times \cdots \times W_k, E, \mathcal{A}]$ as representative: the remaining cases are similar, as the reasoning only depends on the local exchanges of the processes involved in the reduction.

From $\Gamma \vdash_\sigma n[\langle M_1,\dots,M_k\rangle Q \mid R] : T''$ and $\Gamma \vdash n : \rho\mathsf{Amb}[W_1 \times \cdots \times W_k, E, \mathcal{A}]$, by Lemma B.1, we know that $\mathcal{P}(\rho,\sigma,\mathcal{A})$, and there must exist F such that $\mathsf{Pro}[E, {}^\bullet F, \mathcal{A}] \leqslant T''$, and $\Gamma \vdash_\sigma n[\langle M_1,\dots,M_k\rangle Q \mid R] : \mathsf{Pro}[E, {}^\bullet F, \mathcal{A}]$ derives from $\Gamma \vdash_\rho \langle M_1,\dots,M_k\rangle Q \mid R : \mathsf{Pro}[W_1 \times \cdots \times W_k, E, \mathcal{A}]$. From the last judgment, by Lemma B.1.($\mathbf{c_3}$), we know that $\Gamma \vdash_\rho R : \mathsf{Pro}[W_1 \times \cdots \times W_k, E, \mathcal{A}]$ and $\Gamma \vdash_\rho \langle M_1,\dots,M_k\rangle Q : \mathsf{Pro}[W_1 \times \cdots \times W_k, E, \mathcal{A}]$. From the last judgment, by Lemma B.1.($\mathbf{o_1}$), $\Gamma \vdash M_i : W_i$, and (by subsumption) also $\Gamma \vdash_\rho Q : \mathsf{Pro}[W_1 \times \cdots \times W_k, E, \mathcal{A}]$. From this judgment and $\Gamma \vdash_\rho R : \mathsf{Pro}[W_1 \times \cdots \times W_k, E, \mathcal{A}]$ we then derive $\Gamma \vdash_\rho Q \mid R : \mathsf{Pro}[W_1 \times \cdots \times W_k, E, \mathcal{A}]$ by (PAR). From the last judgment, from $\Gamma \vdash n : \rho\mathsf{Amb}[W_1 \times \cdots \times W_k, E, \mathcal{A}]$ and $\mathcal{P}(\rho,\sigma,\mathcal{A})$ one derives $\Gamma \vdash_\sigma n[Q \mid R] : \mathsf{Pro}[E, {}^\bullet F, \mathcal{A}]$ by (AMB), and then $\Gamma \vdash_\sigma n[Q \mid R] : T''$ by subsumption.

From $\Gamma \vdash M_i : W_i$ and $\Gamma, x_1 : W_1,\dots,x_k : W_k \vdash_\sigma P : T'$ which we had derived above, by Lemma B.6, $\Gamma \vdash_\sigma P\{x_i := M_i\} : T'$. Finally, from $\Gamma \vdash_\sigma P\{x_i := M_i\} : T'$ and $\Gamma \vdash_\sigma n[Q \mid R] : T''$, the judgment $\Gamma \vdash_\sigma P\{x_i := M_i\} \mid n[Q \mid R] : T$ derives by the same steps used in Ξ.

5.. The judgment in the hypothesis must have been derived from the judgements $\Gamma \vdash_\sigma (x_1 : W_1,\dots,x_k : W_k)P : T'$, and $\Gamma \vdash_\sigma n[\langle M_1,\dots,M_k\rangle^\uparrow Q \mid R] : T''$ for suitable T' and T''. Let Ξ be the (partial) derivation from these two judgments to the judgment in the hypothesis

From the first judgment, by Lemma B.1.($\mathbf{i_1}$), there exists E such that $\mathsf{Pro}[W_1 \times \cdots \times W_k, {}^? E, \mathcal{A}] \leqslant T'$ and $\Gamma, x_1 : W_1,\dots,x_k : W_k \vdash_\sigma P : \mathsf{Pro}[W_1 \times \cdots \times W_k, {}^? E, \mathcal{A}]$.

From $\Gamma \vdash_\sigma n[\langle M_1,\dots,M_k\rangle^\uparrow Q \mid R] : T''$ and the judgment we just derived, by Lemma B.1.($\mathbf{a_1}$) – ($\mathbf{a_3}$), there exists $H \leqslant W_1 \times \cdots \times W_k$ such that $\mathsf{Pro}[H, {}^\bullet E, \mathcal{A}] \leqslant T''$ and $\Gamma \vdash_\sigma n[\langle M_1,\dots,M_k\rangle^\uparrow Q \mid R] : \mathsf{Pro}[H, {}^\bullet E, \mathcal{A}]$ is derived from either of the sets of hypotheses defined by the cases ($\mathbf{a_1}$) and ($\mathbf{a_2}$). The case ($\mathbf{a_3}$) may be dispensed with, as it implies that a derivation exists for the judgment $\Gamma \vdash_\sigma \langle M_1,\dots,M_k\rangle^\uparrow Q \mid R : \mathsf{Pro}[I, {}^\circ K, \mathcal{B}]$ (for some I, K and $\mathcal{B}$), while such derivation does not exist: if the judgment in question were derivable, then by Lemma B.1.($\mathbf{c_2}$), either $\Gamma \vdash_\sigma \langle M_1,\dots,M_k\rangle^\uparrow Q : \mathsf{Pro}[I, {}^\circ K, \mathcal{B}]$ or $\Gamma \vdash_\sigma \langle M_1,\dots,M_k\rangle^\uparrow Q : \mathsf{Pro}[I, {}^\bullet K, \mathcal{B}]$, would be derivable, contradicting Lemma B.1.($\mathbf{o_3}$).

The cases ($\mathbf{a_1}$) and ($\mathbf{a_2}$) are similar: we prove the second, which is more complex, and leave the first to the reader. The hypotheses are $\Gamma \vdash n : \rho\mathsf{Amb}^\circ[I, H, \mathcal{B}]$ and $\Gamma \vdash_\rho \langle M_1,\dots,M_k\rangle^\uparrow Q \mid R : \mathsf{Pro}[I, {}^\vartriangle H, \mathcal{B}]$. From the last judgment, by Lemma B.1.($\mathbf{c_3}$) (and the reasoning we just made about case ($\mathbf{a_3}$)), it follows that $\Gamma \vdash_\rho \langle M_1,\dots,M_k\rangle^\uparrow Q : T^*$ is derivable with T^* such that $\mathsf{Pro}[I, H, \mathcal{B}] \leqslant T^*$. This, by Lemma B.1.($\mathbf{o_3}$) implies that $H \neq \mathsf{shh}$: from this, since $H \leqslant W_1 \times \cdots \times W_k$, it follows that $H = W_1 \times \cdots \times W_k$. Lemma B.1.($\mathbf{o_3}$) applied to $\Gamma \vdash_\rho \langle M_1,\dots,M_k\rangle^\uparrow Q : T^*$ also implies $\Gamma \vdash M_i : W_i$ and $\Gamma \vdash_\rho Q : T^*$. Then $\Gamma \vdash_\rho Q \mid R : \mathsf{Pro}[I, {}^\vartriangle H, \mathcal{B}]$ derives by the same steps that derived $\Gamma \vdash_\rho \langle M_1,\dots,M_k\rangle^\uparrow Q \mid R : \mathsf{Pro}[I, {}^\vartriangle H, \mathcal{B}]$ from $\Gamma \vdash_\rho \langle M_1,\dots,M_k\rangle^\uparrow Q : T^*$.

Now from $\Gamma \vdash M_i : W_i$ and from $\Gamma, x_1 : W_1,\dots,x_k : W_k \vdash_\rho P : \mathsf{Pro}[W_1 \times \cdots \times W_k, {}^? E, \mathcal{A}]$, which we had derived above, by Lemma B.6, we have $\Gamma \vdash_\sigma P\{x_i := M_i\} : \mathsf{Pro}[W_1 \times \cdots \times W_k, {}^? E, \mathcal{A}]$, and then $\Gamma \vdash_\sigma P\{x_i := M_i\} : T'$ by subsumption. We are ready to conclude:

from $\Gamma \vdash n : \rho\mathsf{Amb}^{\circ}[I,H,\mathcal{B}]$ and $\Gamma \vdash_{\rho} Q \mid R : \mathsf{Pro}[I,{}^{\vartriangle}H,\mathcal{B}]$ we derive $\Gamma \vdash_{\sigma} n[\,Q \mid R\,]$: $\mathsf{Pro}[H,{}^{\bullet}E,\mathcal{A}]$, and then $\Gamma \vdash_{\sigma} n[\,Q \mid R\,] : T''$. From the last judgment and from $\Gamma \vdash_{\sigma} P\{x_i := M_i\} : T'$, one derives $\Gamma \vdash_{\sigma} P\{x_i := M_i\} \mid n[\,Q \mid R\,] : T$ by the steps used in Ξ, $\Box$

LEMMA ASYNCHRONOUS EXCHANGE PRESERVES TYPES.

(1) If $\Gamma \vdash_{\sigma} (x_1 : W_1, \ldots, x_k : W_k)P \mid \langle M_1, \ldots, M_k \rangle : T$, then $\Gamma \vdash_{\sigma} P\{x_i := M_i\} : T$

(2) If $\Gamma \vdash_{\sigma} (x_1 : W_1, \ldots, x_k : W_k)^n P \mid n[\,\langle M_1, \ldots, M_k \rangle \mid Q\,] : T$, then $\Gamma \vdash_{\sigma} P\{x_i := M_i\} \mid n[Q] : T$

(3) If $\Gamma \vdash_{\sigma} \langle M_1, \ldots, M_k \rangle \mid n[\,(x_1 : W_1, \ldots, x_k : W_k)^{\uparrow} P \mid Q\,] : T$, then $\Gamma \vdash_{\sigma} n[P\{x_i := M_i\} \mid Q] : T$

(4) If $\Gamma \vdash_{\sigma} P \mid n[\,\langle M_1, \ldots, M_k \rangle^{\uparrow} Q \mid R\,] : T$ then $\Gamma \vdash_{\sigma} P \mid \langle M_1, \ldots, M_k \rangle \mid n[\,Q \mid R\,] : T$

(5) If $\Gamma \vdash_{\sigma} \langle M_1, \ldots, M_k \rangle^n \mid n[Q] : T$ then $\Gamma \vdash_{\sigma} n[\,\langle M_1, \ldots, M_k \rangle \mid Q\,] : T$

PROOF. The cases 1, 2 and 3 follow by the corresponding cases of Lemma B.7, by (i) choosing **0** as the continuation of the output process, and (ii) observing that $\Gamma \vdash_{\sigma} \langle M_1, \ldots, M_k \rangle : T$ if and only if $\Gamma \vdash_{\sigma} \langle M_1, \ldots, M_k \rangle \mathbf{0} : T$. In case 4 the proof is similar to the corresponding case of Lemma B.7 (in fact, the proof is simpler, and follows without appealing to Lemma B.6). Case 5 is left to the reader.

LEMMA SUBJECT CONGRUENCE.

(1) If $\Gamma \vdash_{\sigma} P : T$ and $P \equiv Q$ then $\Gamma \vdash_{\sigma} Q : T$.
(2) If $\Gamma \vdash_{\sigma} P : T$ and $Q \equiv P$ then $\Gamma \vdash_{\sigma} Q : T$.

PROOF. By simultaneous induction on the depths of the derivations of $P \equiv Q$ and $Q \equiv P$.

(1) If $\Gamma \vdash_{\sigma} P : T$ and $P \equiv Q$ then $\Gamma \vdash_{\sigma} Q : T$.
(Struct Refl) The hypothesis is $P \equiv P$, and the proof follows directly from the assumption $\Gamma \vdash_{\sigma} P : T$.
(Struct Symm) Then $P \equiv Q$ derives from $Q \equiv P$. $\Gamma \vdash_{\sigma} Q : T$ follows by induction hypothesis **(2)**.
(Struct Trans) Then $P \equiv Q$ derives from $P \equiv R$ and $R \equiv Q$ for some R. From $P \equiv R$ and $\Gamma \vdash_{\sigma} P : T$, by induction hypothesis **(1)** $\Gamma \vdash_{\sigma} R : T$. From the last judgment, and from $R \equiv Q$, again by induction hypothesis **(1)** $\Gamma \vdash_{\sigma} Q : T$.
(Struct Par Assoc) We prove this case as representative of **(Struct Par Dead)** and **(Struct Par Comm)**. The hypotheses $\Gamma \vdash_{\sigma} P \mid (Q \mid R) : T$ and $P \mid (Q \mid R) \equiv (P \mid Q) \mid R$. By Lemma B.1, there exist E and F such that $\mathsf{Pro}[E,{}^{?}F,\mathcal{A}] \leqslant T$ and $\Gamma \vdash_{\sigma} P \mid (Q \mid R)$: $\mathsf{Pro}[E,{}^{?}F,\mathcal{A}]$. Also, we may safely focus on derivations from the assumptions defined by cases ($\mathbf{c_1}$)–($\mathbf{c_3}$).
($\mathbf{c_1}$). $T \leqslant \mathsf{Pro}[E,F,\mathcal{A}]$: By two further applications of Lemma B.1.($\mathbf{c_1}$), we know that the judgment in the hypothesis is derivable from $\Gamma \vdash_{\sigma} P : T$, $\Gamma \vdash_{\sigma} Q : T$ and $\Gamma \vdash_{\sigma} R : T$. Then the judgment $\Gamma \vdash_{\sigma} P \mid (Q \mid R) : T$ follows by two applications of (PAR).
($\mathbf{c_2}$). $T = \mathsf{Pro}[E,{}^{\circ}F,\mathcal{A}]$. By two applications of Lemma B.1.($\mathbf{c_2}$), we can focus on the following cases:

– $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[E,{}^{\circ}F,\mathcal{A}]$, $\Gamma \vdash_{\sigma} Q : \mathsf{Pro}[E,{}^{\bullet}F,\mathcal{A}]$ and $\Gamma \vdash_{\sigma} R : \mathsf{Pro}[E,{}^{\bullet}F,\mathcal{A}]$. To derive the judgment $\Gamma \vdash_{\sigma} (P \mid Q) \mid R : T$, apply (PAR $\circ$) twice and then subsumption.
– $\Gamma \vdash_{\sigma} P : \mathsf{Pro}[E,{}^{\bullet}F,\mathcal{A}]$, $\Gamma \vdash_{\sigma} Q : \mathsf{Pro}[E,{}^{\circ}F,\mathcal{A}]$ and $\Gamma \vdash_{\sigma} R : \mathsf{Pro}[E,{}^{\bullet}F,\mathcal{A}]$. As above, apply (PAR $\circ$) twice and then subsumption.

– $\Gamma \vdash_\sigma P : \mathsf{Pro}[E, {}^\bullet F, \mathcal{A}]$, $\Gamma \vdash_\sigma Q : \mathsf{Pro}[E, {}^\bullet F, \mathcal{A}]$ and $\Gamma \vdash_\sigma R : \mathsf{Pro}[E, {}^\circ F, \mathcal{A}]$. Apply (PAR $\bullet$) and then (PAR $\circ$), followed by subsumption.

($\mathbf{c_3}$). $T = \mathsf{Pro}[E, {}^\vartriangle F, \mathcal{A}]$. The proof is similar to the previous case, with more subcases to consider, but no further difficulty.

(Struct Res Dead) The hypothesis is $(\nu x{:}W)\mathbf{0} \equiv \mathbf{0}$. The judgment $\Gamma \vdash_\sigma (\nu x{:}W)\mathbf{0} : T$ must have been derived from $\Gamma, x{:}W \vdash_\sigma \mathbf{0} : T'$, for $T' \leqslant T$ (and $W = \sigma\mathsf{Amb}^?[E,F,\mathcal{A}]$ for some E, F and $\mathcal{A}$). Thus $\Gamma, x{:}W \vdash \diamond$ and hence Lemma B.1.3 and B.1.2 yield $\Gamma \vdash \diamond$. The judgment $\Gamma \vdash_\sigma \mathbf{0} : T$ derives now by (DEAD).

(Struct Res Res) The hypothesis is $(\nu x{:}W)(\nu y{:}W')P \equiv (\nu y{:}W')(\nu x{:}W)P$, and $\Gamma \vdash_\sigma (\nu x{:}W)(\nu y{:}W')P : T$ must have been derived from $\Gamma, x{:}W, y : W' \vdash_\sigma P : T'$, with $T \leqslant T$. From the last judgment, by two applications of the rule (NEW), followed by subsumption, we derive $\Gamma \vdash_\sigma (\nu y{:}W')(\nu x{:}W)P : T$ as desired.

(Struct Res Par) The hypothesis is $(\nu x{:}W)(P \mid Q) \equiv P \mid (\nu x{:}W)Q$ with $x \notin fn(P)$. The judgment $\Gamma \vdash_\sigma (\nu x{:}W)(P \mid Q) : T$ must have been derived from $\Gamma, x{:}W \vdash_\sigma (P \mid Q) : T^*$ with $T^* \leqslant T$. The last judgment, in turn, must have been derived from $\Gamma, x{:}W \vdash_\sigma P : T'$ and $\Gamma, x{:}W \vdash_\sigma Q : T''$ for suitable T' and T''. From the first of the two judgments, by the Lemma B.5.3 one deduces $\Gamma \vdash_\sigma P : T'$. From the second, by (NEW) $\Gamma \vdash_\sigma (\nu x{:}W)Q : T''$. Now $\Gamma \vdash_\sigma P \mid (\nu x{:}W)Q : T$ by the suitable rule for parallel composition.

(Struct Res Amb) The hypothesis is $(\nu x{:}W)M[\![P]\!] \equiv M[\![(\nu x{:}W)P]\!]$ with $x \notin fn(M)$. The judgment $\Gamma \vdash_\sigma (\nu x{:}W)M[\![P]\!] : T$ must be derived from $\Gamma, x{:}W \vdash_\rho P : T'$, for a suitable T', and from $\Gamma, x{:}W \vdash M : \rho\mathsf{Amb}^?[E,F,\mathcal{A}]$. Let Ξ be the typing derivation from these two judgments to the judgment in the hypothesis. Since $x \notin fn(M)$, Lemma B.1.3 applied to the first judgment implies that $\Gamma \vdash M : \rho\mathsf{Amb}^?[E,F,\mathcal{A}]$. From the second judgment $\Gamma \vdash_\rho (\nu x{:}W)P : T'$ derives by (NEW). Now $\Gamma \vdash_\sigma M[\![(\nu x{:}W)P']\!] : T$ derives by the same steps used in Ξ.

(Struct Path Assoc) The hypotheses are $(M.M').P \equiv M.(M'.P)$ and $\Gamma \vdash_\sigma (M.M').P : T$. We have to distinguish two cases depending on the type T, as informed by Lemma B.1:

($\mathbf{p_1}$). $T \leqslant \mathsf{Pro}[E,F,\mathcal{A}]$: in this case $\Gamma \vdash_\sigma P : T$ and $\Gamma \vdash M.M' : \sigma\mathsf{Cap}[F,\mathcal{A}]$. The last judgment must have been derived from $\Gamma \vdash M' : \sigma\mathsf{Cap}[F,\mathcal{A}]$ and $\Gamma \vdash M' : \sigma\mathsf{Cap}[F,\mathcal{A}]$ by the rule (PATH). Then, by two applications of (PREFIX), we have that $\Gamma \vdash_\sigma M.(M'.P) : T$ is derivable, as desired.

($\mathbf{p_2}$). $T = \mathsf{Pro}[E, {}^\circ F, \mathcal{A}]$ or $T = \mathsf{Pro}[E, {}^\vartriangle F, \mathcal{A}]$. We have the following subcases:

– $\Gamma \vdash\!\circ\, M.M' : \rho\mathsf{Cap}[G,\mathcal{B}]$ and $\Gamma \vdash_\sigma P : \mathsf{Pro}[E, {}^\circ F, \mathcal{A}]$. The first judgment must have been derived by (POLYPATH) from $\Gamma \vdash\!\circ\, M : \tau\mathsf{Cap}[H,\mathcal{C}]$, for some H, $\mathcal{C}$, and τ, and from $\Gamma \vdash\!\circ\, M' : \rho\mathsf{Cap}[G,\mathcal{B}]$. Then $\Gamma \vdash_\sigma M.(M'.P) : \mathsf{Pro}[E, {}^\circ F, \mathcal{A}]$ derives by two applications of (PREFIX $\circ$).

– $\Gamma \vdash\!\circ\, M.M' : \sigma\mathsf{Cap}[F,\mathcal{A}]$ and $\Gamma \vdash_\sigma P : \mathsf{Pro}[E, {}^\vartriangle F, \mathcal{A}]$. The first judgment must have been derived by (POLYPATH) from $\Gamma \vdash\!\circ\, M : \rho\mathsf{Cap}[H,\mathcal{B}]$ and $\Gamma \vdash\!\circ\, M' : \sigma\mathsf{Cap}[F,\mathcal{A}]$. Then $\Gamma \vdash_\sigma M.(M'.P) : \mathsf{Pro}[E, {}^\circ F, \mathcal{A}]$ derives by (PREFIX $\vartriangle$) and (PREFIX $\circ$).

A final subsumption step derives $\Gamma \vdash_\sigma M.(M'.P) : \mathsf{Pro}[E, {}^\vartriangle F, \mathcal{A}]$, in both cases.

(Struct Repl) The hypothesis is $!P \equiv P \mid\, !P$. The judgment $\Gamma \vdash_\sigma\, !P : T$ must have been derived from $\Gamma \vdash_\sigma P : T'$ with $T' \leqslant \mathsf{Pro}[E,F,\mathcal{A}] \leqslant T$. If $T' = \mathsf{Pro}[E,F,\mathcal{A}]$, one derives $\Gamma \vdash_\sigma\, !P : T'$ by (REPL) and then $\Gamma \vdash_\sigma P \mid\, !P : T$ by (PAR) followed by subsumption. If $T' = \mathsf{Pro}[E, {}^\bullet F, \mathcal{A}]$, one derives $\Gamma \vdash_\sigma\, !P : T'$ by (REPL $\bullet$) and then $\Gamma \vdash_\sigma P \mid\, !P : T$ by (PAR $\bullet$) followed by subsumption.

(Struct Cong Par) The hypothesis is $P \mid R \equiv Q \mid R$ derived from $P \equiv Q$. The judgment $\Gamma \vdash_\sigma P \mid R : T$ must have been derived from $\Gamma \vdash_\sigma P : T'$ and $\Gamma \vdash_\sigma R : T''$ for suitable T' and T''. Let Ξ the partial derivation from these two judgments to $\Gamma \vdash_\sigma P \mid R : T$. By the induction hypothesis **(1)** on $P \equiv Q$, one has $\Gamma \vdash_\sigma Q : T'$. Then $\Gamma \vdash_\sigma Q \mid R : T$ derives by the same steps used in Ξ.

The remaining congruence cases, namely **(Struct Cong Action)**, **(Struct Cong Agent)**, **(Struct Cong Input)**, **(Struct Cong Output)**, **(Struct Cong New)**, **(Struct Cong Repl)** are all proved similarly to the previous case.

(2) If $\Gamma \vdash_\sigma P : T$ and $Q \equiv P$ then $\Gamma \vdash_\sigma Q : T$.
The proof follows by the same analysis of case 1. In case **(Struct Symm)**, we use the induction hypothesis **(1)**, in place of **(2)**. In all the remaining cases, we use the induction hypothesis **(2)** in place of **(1)**. The only nontrivial cases are **(Struct Par Assoc)**, which is symmetric to the corresponding subcase of 1, and **(Struct Path Assoc)**, which we give below.

(Struct Path Assoc) The hypotheses are $M.(M'.P) \equiv (M.M').P$ and $\Gamma \vdash_\sigma M.(M'.P) : T$. We have to distinguish two cases depending on the type T, as informed by Lemma B.1:

($\mathbf{p_1}$). $T \leqslant \mathsf{Pro}[E,F,\mathcal{A}]$: in this case $\Gamma \vdash M : \sigma\mathsf{Cap}[F,\mathcal{A}]$ and $\Gamma \vdash_\sigma M'.P : T$. By Lemma B.1.($\mathbf{p_1}$), applied to the judgment $\Gamma \vdash_\sigma M'.P : T$, it follows that $\Gamma \vdash M' : \sigma\mathsf{Cap}[F,\mathcal{A}]$ and $\Gamma \vdash_\sigma P : T$. Then $\Gamma \vdash M.M' : \sigma\mathsf{Cap}[F,\mathcal{A}]$ derives from (PATH), and $\Gamma \vdash_\sigma (M.M').P : T$ by (PREFIX).

($\mathbf{p_2}$). $T = \mathsf{Pro}[E,{}^\circ F,\mathcal{A}]$ or $T = \mathsf{Pro}[E,{}^\vartriangle F,\mathcal{A}]$. We have two sub-cases.

In the first sub-case, one has $\Gamma \vdash^{\circ} M : \rho\mathsf{Cap}[G,\mathcal{B}]$ and $\Gamma \vdash M'.P : \mathsf{Pro}[E,{}^\circ F,\mathcal{A}]$. By Lemma B.1.($\mathbf{p_1}$), applied to the judgment $\Gamma \vdash_\sigma M'.P : \mathsf{Pro}[E,{}^\circ F,\mathcal{A}]$, it follows that either $\Gamma \vdash^{\circ} M' : \tau\mathsf{Cap}[H,\mathcal{C}]$ and $\Gamma \vdash_\sigma P : \mathsf{Pro}[E,{}^\circ F,\mathcal{A}]$, or $\Gamma \vdash^{\circ} M' : \sigma\mathsf{Cap}[F,\mathcal{A}]$ and $\Gamma \vdash_\sigma P' : \mathsf{Pro}[E,{}^\vartriangle F,\mathcal{A}]$. In the first case apply (POLYPATH) to derive $\Gamma \vdash M.M' : \tau\mathsf{Cap}[H,\mathcal{C}]$, and then (PREFIX $\circ$). In the second case, apply (POLYPATH) to derive $\Gamma \vdash M.M' : \sigma\mathsf{Cap}[F,\mathcal{A}]$, and then (PREFIX $\vartriangle$).

In the second sub-case, $\Gamma \vdash^{\circ} M. : \sigma\mathsf{Cap}[F,\mathcal{A}]$ and $\Gamma \vdash_\sigma M'.P : \mathsf{Pro}[E,{}^\vartriangle F,\mathcal{A}]$, and the proof is similar to the one just given.

LEMMA SUBJECT CONGRUENCE: THE ASYNCHRONOUS CASE. Take $\eta \neq \uparrow$. Then, $\Gamma \vdash \langle M\rangle^\eta P : T$ if and only if $\Gamma \vdash \langle M\rangle^\eta \mid P : T$, where both judgements are derived in the system of Section 7.

PROOF. The proof follows almost directly by an inspection of the typing rules.

THEOREM TYPE PRESERVATION FOR SYNCHRONOUS REDUCTION. If $\Gamma \vdash_\sigma P : T$ and $P \longrightarrow Q$ then $\Gamma \vdash_\sigma Q : T$.

PROOF. By induction on the derivation of $P \longrightarrow Q$. The cases of top-level reduction follow by Lemmas B.3, B.4 and B.7. The inductive cases follow directly by the induction hypothesis, with an additional appeal to Lemma B.9 for the case of reduction via structural congruence. ☐

THEOREM TYPE PRESERVATION FOR TAGGED ASYNCHRONOUS REDUCTION.
Given a type environment Γ, say that a security assignment γ is Γ-*consistent* if and only if for all $x \in dom(\Gamma)$, $\Gamma(x) = \sigma\mathsf{Amb}^?[\ldots]$ implies $\gamma(x) = \sigma$. Now assume that $\Gamma \vdash_\sigma P : T$. Then for every Γ-consistent assignment γ and process Q such that $P \longrightarrow_{(\sigma,\gamma)} Q$, we have $\Gamma \vdash_\sigma Q : T$.

PROOF. By induction on the derivation of $P \longrightarrow_{(\sigma,\gamma)} Q$. The cases of top-level move reductions follow by Lemmas B.3, B.4. The cases of communication follows by Lemma B.8. As for the inductive steps, the proof if guided by the inductive cases of definition of the tagged reduction (cf. Section 9.1):

– case (PAR) follows by the inductive hypothesis and an application of the appropriate (PAR ?) rule
– case (STRUCT) follows by the induction hypothesis and Lemma B.9.
– case (NEW) follows again by the induction hypothesis, as the side condition to the rule guarantees that the assignment $\gamma,(n:\rho)$ is $(\Gamma,n:A)$-consistent.
– case (AMB) also follows by the induction hypothesis, as γ being Γ-consistent implies that $\Gamma \vdash_\sigma a[P] : T$ depends on $\Gamma \vdash_{\gamma(a)} P : T'$ for a suitable T'.

THEOREM TYPE SOUNDNESS. If $\Gamma \vdash_\sigma P : T$, then for every Γ-consistent security assignment, and process Q such that $P \longrightarrow^*_{(\sigma,\gamma)} Q$ we have $Q \not\longrightarrow_{(\sigma,\gamma)} \mathsf{err}$.

PROOF. We first show that $P \not\longrightarrow_{(\sigma,\gamma)} \mathsf{err}$: to show that, it is more convenient to prove the contrapositive, namely that $P \longrightarrow_{(\sigma,\gamma)} \mathsf{err}$ implies $\Gamma \not\vdash_\sigma P : T$. The proof proceeds by induction on the derivation of $P \longrightarrow_{(\sigma,\gamma)} \mathsf{err}$.

– For the basis of induction, we assume, for the purpose of contradiction, that $P \longrightarrow_{(\sigma,\gamma)} \mathsf{err}$ and $\Gamma \vdash_\sigma P : T$. Then we have four possible cases: we give the proof of the case (ERR OUTPUT $\uparrow$) as representative.
  In this case, the redex has the form $P \mid n[\langle M\rangle^{\uparrow} Q \mid R]$ and the reduction to err implies that $\neg\mathcal{P}(\gamma(n),\sigma,\mathsf{w})$. Since γ is Γ-compatible, it must be the case that $\Gamma \vdash \gamma(n)\mathsf{Amb}^?[\_,\_]$ for suitable types and access modes. From our hypothesis $\Gamma \vdash_\sigma P \mid n[\langle M\rangle^{\uparrow} Q \mid R] : T$ by repeated applications of LemmaB.1, it follows that the derivation depends on the side-condition $\mathcal{P}(\gamma(n),\sigma,\mathsf{w})$ to the (AMB) or the (AMB $\triangle$) rule yielding the sought contradiction (that the judgment is derived by either of these rules, and not by (AMB $\circ$) follows by the same reasoning as in case 5 of Lemma B.7).
– For the inductive steps we have to consider several cases. If $P \longrightarrow_{(\sigma,\gamma)} \mathsf{err}$ by (ERR STRUCT), then there exists Q such that $P \equiv Q$ and $Q \longrightarrow_{(\sigma,\gamma)} \mathsf{err}$. By the induction hypothesis, this implies $\Gamma \not\vdash_\sigma Q : T$, and then proof follows by Lemma B.9. The cases in which $P \longrightarrow_{(\sigma,\gamma)} \mathsf{err}$ by any of the contextual reductions follow directly by the induction hypothesis, noting that a process term is well-typed if and only if so are all of its sub-terms.

The proof of the theorem follows now by subject reduction. From $\Gamma \vdash_\sigma P : T$ and from $P \longrightarrow (\sigma,\gamma)^* Q$, by Theorem B.12 we know that $\Gamma \vdash_\sigma Q : T$, and we have just proved that this implies that $Q \not\longrightarrow_{(\sigma,\gamma)} \mathsf{err}$.