Mobile Safe Ambients

FRANCESCA LEVI
University of Pisa
and
DAVIDE SANGIORGI
INRIA Sophia-Antipolis

Two forms of interferences are individuated in Cardelli and Gordon's *Mobile Ambients* (MA): *plain interferences*, which are similar to the interferences one finds in CCS and π-calculus; and *grave interferences*, which are more dangerous and may be regarded as programming errors. To control interferences, the MA movement primitives are modified; the resulting calculus is called *Mobile Safe Ambients* (SA).

The modification has also computational significance. In the MA interaction rules, an ambient may enter, exit, or open another ambient. The second ambient undergoes the action; it has no control on *when* the action takes place. In SA this is rectified: any movement takes place only if both participants agree.

Existing type systems for MA can be easily adapted to SA. The type systems for controlling mobility, however, appear to be more powerful in SA, in that (i) type systems for MA may give more precise information when transplanted onto SA, and (ii) new type systems may be defined. Two type systems are presented that remove all grave interferences.

Other advantages of SA are: a useful algebraic theory; programs sometimes more robust (they require milder conditions for correctness) and/or simpler. All these points are illustrated on several examples.

Categories and Subject Descriptors: []:

General Terms:

Additional Key Words and Phrases:

1. MOTIVATIONS

The calculus of *Mobile Ambients* [Cardelli and Gordon 1998a] (MA) is proposed both as a core programming language for the Web and as a model for reasoning about properties of mobile processes, including security. In contrast with previous formalisms for mobile processes such as the π-calculus, whose computational model is based on the notion of *communication*, the MA computational model is based on the notion of *movement*. An ambient, which may be thought of as a named location, is the unit of movement. Processes within the same ambient may exchange messages; ambients may be nested, so to form a hierarchical structure. The three primitives for movement allow: an ambient to enter another ambient, $n[\, \mathtt{in}\, m. P \mid Q\,] \mid m[\, R\,] \longrightarrow m[\, n[\, P \mid Q\,] \mid R\,]$; an ambient to exit another ambient,

This is an expanded and revised version of the paper 'Controlling Interference in Ambients', originally appearing in *Proc. of Principles of Programming Language*, 2000.
Author's address: Francesca Levi, Dipartimento di Informatica, University of Pisa, Corso Italia, 40, Pisa, Italy; Davide Sangiorgi, INRIA, 2004, Rue des Lucioles, Sophia Antipolis, France
Acknowledgement of support: Research partially funded by the european project FET-Global computing "PROFUNDIS".

$m[\, n[\, \mathtt{out}\, m.\, P \mid Q\,] \mid R\,] \longrightarrow n[\, P \mid Q\,] \mid m[\, R\,]$; a process to dissolve an ambient boundary thus obtaining access to its content, $\mathtt{open}\, n.\, P \mid n[\, Q\,] \longrightarrow P \mid Q$. We call the pairs of ambients or processes involved in a reduction a *redex*, and the corresponding reduction the *contraction* of the redex. Elegant type systems for MA have been given; they control the type of values exchanged within an ambient [Cardelli and Gordon 1999c] and the mobility of ambients [Cardelli et al. 1999].

We find however three aspects of the MA model unsatisfactory. They have to do with interferences, algebraic laws, and passivity of ambients involved in a movement. We discuss these aspects—strongly related with each other—below.

Our work began with a study of *interference* in MA. In concurrency, an interference occurs when the activity of a process is damaged or corrupted because of the activities of other processes. Interference is one of the most tricky and subtle phenomena in concurrency, and may make both the activities of programming and of reasoning on programs hard. In MA, the most important interference is *redex interference*[1]. It is caused by two, or more, redexes sharing one of the interacting partners. Here are some examples:

$$n[\, \mathtt{in}\, m.\, P\,] \mid m[\, Q\,] \mid m[\, R\,] \tag{1}$$

$$\mathtt{open}\, n.\, P \mid \mathtt{open}\, n.\, Q \mid n[\, R\,] \tag{2}$$

In (1), the redex formed by $n[\, \mathtt{in}\, m.\, P\,]$ and $m[\, Q\,]$ interferes with that formed by $n[\, \mathtt{in}\, m.\, P\,]$ and $m[\, R\,]$. The contraction of one of the redexes destroys the other one. The same happens in (2). We find similar interferences in calculi for concurrency such as CCS and π-calculus [Milner 1989; Milner et al. 1992; Sangiorgi and Walker 2001]. For instance, in the π-calculus process

$$\overline{x}z.\, P \mid x(y).\, Q \mid x(y).\, R \tag{3}$$

the output $\overline{x}z$ may be consumed either by input $x(y).\, Q$ or by $x(y).\, R$. We call interferences such as (1-3) *plain interferences.* They occur when a process may engage the "same" interaction with two different partners. Intuitively, a process requests or offers a service and finds at least two processes willing to react. Plain interferences are sometimes desirable, for instance to program useful non-deterministic behaviours. An example is a request of printing a job that is non-deterministically served by one out of several printers.

In CCS or π-calculus[2] all redex interferences are plain interferences. In MA, however, we find a new, and much more dangerous, form of interference, which we call *grave interference.* Consider the processes

$$\mathtt{open}\, n \mid n[\, \mathtt{in}\, m.\, P\,] \mid m[\, Q\,] \tag{4}$$

$$h[\, n[\, \mathtt{in}\, m.\, P \mid \mathtt{out}\, h.\, R\,] \mid m[\, Q\,]\,] \tag{5}$$

$$h[\, Q\,] \mid n[\, \mathtt{in}\, h \mid m[\, \mathtt{out}\, n.\, P\,]\,] \tag{6}$$

[1]Another source of interferences is aliasing, similar to that one finds in imperative programs [Reynolds 1978]; but the problems causes by aliasing interference often show up as forms of redex interference.

[2]We are thinking at the sum-free calculi, since MA has no sum or other operators for expressing a choice between two possible behaviours.

In (4), n may be opened, but may also jump into m; in (5) n may move out of the parent h, but also into the sibling m; in (6), n can let m go out of it, or jump into the sibling h. As in (1-2), so in (4-6) the same ambient n is shared by two different redexes. But, in contrast with the previous examples, here the shared ambient may perform logically different interactions. The choice of contraction between the two redexes has dramatic consequences on n: in (4) n is either dissolved or will find refuge inside m; in (5) n will end up either outside h or inside a son of h. In (4) and (5) the contraction of a redex destroys the other one; in (6) the second redex remains, but is corrupted (i.e., it will produce an unexpected outcome when later contracted). The capability `out` can be exercised both before and after the $\mathtt{in}\, h$ capability; but one of the results can be quite upsetting for m. The systems (4-6) are, from n's point of view, inconsistent.

One should bear in mind that movement in MA is *local*—an ambient may only interact with neighbouring ambients. In an ambient system, the spatial configuration of ambients gives us a labeled tree, in which the subtrees are the subambients and the labels of the edges are the ambient names. A process in the system has therefore a global address, given by the path of ambient names from the root to the position of the process. Locality of movement implies that a process programmed to accomplish a certain task at a given address is probably nonsense at a different address. Now, the systems (4-6) can evolve in two different ways, and a consequence, some processes (P and R) can be executed at two places with different global addresses. It is very likely that these processes will not work correctly for one of the addresses. This situation does not arise with plain interferences: although the systems in (1-2) have different evolutions, the processes affected by the movements end up in places with the *same* global address.

Intuitively, a plain interference originates from reductions whose redexes overlap, the overlapping indicating a competition for the same resource. By contrast, grave interferences originate from *non-overlapping redexes*, which indicates that the different reductions represent logically different activities. (The formalisation of this idea in MA is however complex, see below.) Interfering non-overlapping redexes may make distributed implementations complex. Two terms ready and willing to interact may have to synchronise with processes possibly residing at different sites, because the interaction of the two terms could interfere with that of other (separate) terms.

The existence of grave interferences has other serious consequences:

(1) the algebraic theory is poor;
(2) it is difficult to write correct programs, that is, programs that behave in the expected way in *all* contexts;
(3) it is difficult to prove behavioural properties of processes.

The MA paper series present several examples of modelisation of entire calculi or specific systems. But (with the exception of the firewall-crossing protocol [Cardelli and Gordon 1999a]) none of the examples has been proved correct. Only simulations of the reductions of the source processes by the corresponding MA processes are given. While such simulations may be satisfactory for sequential languages, they are not in concurrency, for they say nothing about possible other reductions that the target processes may take. In Section 8 we shall indeed suggest that for some of the examples strong conditions on the surrounding contexts are necessary for correctness.

We tried to develop an algebraic theory for MA, but failed (even taking into account causality or type information) precisely because of the above interference issues. Techniques for proving behavioural properties and laws on MA are proposed in [Cardelli and Gordon 1999a; 1999b] but, aside structural laws, the only laws given are $\boldsymbol{\nu} n(n[\,P\,]) = \mathbf{0}$ and $\boldsymbol{\nu} n(n[\,\mathbf{0}\,] \mid \mathtt{open}\, n.\, P) = P$ (in both cases n should not be free in P); and, although the laws are simple, their proofs are fairly involved. Especially difficult in MA is the formulation of laws that relate the reduction relation to behavioural equality. These laws are at the heart of the algebraic theory of any calculus.

In our opinion, the occurrence of grave interferences should be regarded as a programming error. Unfortunately, finding conditions that guarantee absence of grave interferences in a process and all its derivatives, without imposing draconian restrictions on the calculus, appears hard. Type systems for controlling mobility [Cardelli et al. 1999], for instance, are too weak.

The third problem of MA concerns the form of its movement rules. The rules allow an ambient to enter, to exit, or to open another ambient. The second ambient undergoes the action. No permission is required from this ambient, which is not even aware that the movement took place. As a consequence, it may be hard, for instance, to program causal dependencies among actions, or to control the resources available inside a given ambient.

Names and capabilities do represent a form of permission (a process cannot open ambients of unknown name, for instance). However, since both names and capabilities are data, no control on when and how they are used is possible once they are given away.

To remedy the above problems while maintaining the essence of the computational model, we propose a modification of the MA primitives. We call *Mobile Safe Ambients* (SA) the new calculus. In SA, any movement (in, out, open) takes place only if *both* participants agree. This is achieved by means of the *coactions* $\overline{\mathtt{in}}\, n, \overline{\mathtt{out}}\, n, \overline{\mathtt{open}}\, n$, whose addition is the main difference from the syntax of MA. The basic movement rules thus become:

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

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

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

The modification has computational significance. In SA, an ambient n that wishes to move into, or out from, an ambient m needs the collaboration of m to achieve the movement. The collaboration can be thought of as an explicit permission. Further, since m is aware of the movement, it can also decide appropriate actions in response to it. For instance, m may decide to move into another ambient h *only after* n has entered m. The case of open operation is even more striking. In MA, a process can arbitrarily decide to open an ambient m of which the process knows the name. In SA, by contrast, the process needs collaboration from m, which, using the coaction $\overline{\mathtt{open}}\, m$, decides if and when the operation occurs.

Grave interferences also exist in SA, but they are easier to spot and to control. In

Section 3 we show a simple definition of interference in SA, in which the grave interferences are precisely the interferences with non-overlapping redexes. (It should be possible to adapt these definitions to MA, but they would become rather complex.) Then, in Section 4, we propose type systems for SA that allow us to remove all grave interferences.

The modifications that yield SA do not reduce the expressiveness of the model. In some cases the SA programs are actually simpler. An example is the final encoding of channels in Section 8: we can write in SA the original encoding of channels into MA [Cardelli and Gordon 1999c; Cardelli et al. 1999], but also a new one that is simpler and, in contrast with the original one, is purely compositional. Even when the programs are similar, the SA program may require fewer conditions for correctness; examples are the protocols for renaming, for routable packets and the firewall protocol in Section 8. Existing type systems for MA can be easily adapted to SA. The type systems for controlling mobility, however, appear to be more powerful in SA, in that (i) type systems for MA may give more precise information when transplanted onto SA (see for instance the discussion in Example 4.2), and (ii) new type systems may be defined. The same happens for other forms of static analysis, such as control flow analysis. In Section 7 we formulate a sample of algebraic laws, some of which rely on the type systems previously introduced. As an indication of the power of the laws, we use them to prove, purely algebraically, correctness properties for several examples.

2. MOBILE SAFE AMBIENTS: SYNTAX AND SEMANTICS

The syntax of *Mobile Safe Ambients* (SA) is presented in Table I. In calculi for mobility, infinite behaviours are described using either the replication construct $!P$, which represents an unbounded number of copies of P in parallel, or the recursion construct $\mathtt{rec}\,X.\,P$. In MA one usually finds the former; for SA we prefer the latter, which is more convenient for the typing rules of Section 4. To have simpler typing rules, we require that in $\mathtt{rec}\,X.\,P$ the variable X does not appear free inside ambients in the body P. We use $!P$ as an abbreviation for $\mathtt{rec}\,X.\,(P \mid X)$. Expressions that are not variables or names are called *capabilities*; actions and coactions are the *simple capabilities*. We often omit the trailing $\mathbf{0}$ in processes $M.\,\mathbf{0}$, and we omit type annotations in restrictions and abstractions whenever not important or clear from the context. Parallel composition has the least syntactic precedence, thus $m[\,M.\,P \mid Q\,]$ reads $m[\,(M.\,P) \mid Q\,]$. A restriction $(\boldsymbol{\nu} n : W)\,P$ is a binder for name n with scope P; similarly an abstraction $(x : W)P$ is a binder for variables x with scope P, and $\mathtt{rec}\,X.\,P$ is a binder for X with scope P. These binders give rise to the definitions of free and bound names, variables, and recursion variables in the usual way. We write $\mathrm{fn}(P)$ and $\mathrm{fv}(P)$ for the free names and free variables of process P; we write $\mathrm{n}(P)$ for the union of the free and bound names of P. We apply a similar notation for names and variables of expressions. We identify processes and expressions that are alpha-convertible, that is, that can be made syntactically equal by a change of bound names and bound variables. A process is *closed* if it has no free variables and recursion variables. We say that a process is *closed on recursion variables* if it has no free recursion variables. Except for some results in Sections 4 and 5, on types, throughout the paper all the results we present are on processes that are closed on recursion variables.

Table I. The syntax of Mobile Safe Ambients

$h, k, \ldots n, m$	*Names*	x, y, z	*Variables*
		X, Y, Z	*Recursion variables*
	Expressions		*Processes*
$M, N \;:=\; x$	*(variable)*	$P, Q, R \;:=\; \mathbf{0}$	*(nil)*
$\mid\; n$	*(name)*	$\mid\; P \mid Q$	*(parallel)*
$\mid\; \mathtt{in}\, M$	*(enter)*	$\mid\; (\boldsymbol{\nu} n : W)\, P$	*(restriction)*
$\mid\; \overline{\mathtt{in}}\, M$	*(allow enter)*	$\mid\; M.\, P$	*(prefixing)*
$\mid\; \mathtt{out}\, M$	*(exit)*	$\mid\; M[\, P\,]$	*(ambient)*
$\mid\; \overline{\mathtt{out}}\, M$	*(allow exit)*	$\mid\; \langle M \rangle$	*(message)*
$\mid\; \mathtt{open}\, M$	*(open)*	$\mid\; (x : W)P$	*(abstraction)*
$\mid\; \overline{\mathtt{open}}\, M$	*(allow open)*	$\mid\; X$	*(recursion variable)*
$\mid\; M.\, N$	*(path)*	$\mid\; \mathtt{rec}\, X.\, P$	*(recursive process)*
$\mid\; \epsilon$	*(empty path)*		

The one-step reduction relation $\longrightarrow$ of SA is defined in the usual way. It exploits the auxiliary relation $\equiv$ of structural congruence (Table III) to bring the participants of a potential interaction into contiguous positions; the definition of $\equiv$ is standard, and includes rules for commuting the positions of parallel components, for stretching the scope of a restriction, for unfolding recursion. Besides the use of recursion in place of replication, the main differences with the reduction semantics of MA are the basic movement rules. Another difference is that we prefer to view the consumption of an ϵ prefix as a reduction rule (R-eps), rather than a structural axiom; this is also simplifies some technical results (for instance, Lemma 5.7). The full set of reduction rules is in Table II, including the axiom that allows the exchange of messages. Both in MA and in SA, communication is asynchronous, takes place inside ambients, and is anonymous—it does not use channel or process names. Structural congruence is defined on all processes, whereas the reduction relation is defined on the closed processes.
We write $\Longrightarrow$ for the reflexive and transitive closure of $\longrightarrow$.

3. INTERFERENCES

We formalise the intuitive idea of interferences, from Section 1, using a refined semantics of SA. We only take into account the interferences resulting from movement of ambients; those resulting from communication are orthogonal and, since they are similar to those of value-passing process calculi such as CCS, do not give new problems.

We assume a countable infinite set of colors. A colored process is a process in which every simple capability that is ready to fire has a distinct color. Formally, using χ to range over colors, *Colored SA* (CSA) is defined by adding the following productions to those of the grammars of Table I:

$$M := \mathtt{in}\, M_\chi \;\Big|\; \overline{\mathtt{in}}\, M_\chi \;\Big|\; \mathtt{out}\, M_\chi \;\Big|\; \overline{\mathtt{out}}\, M_\chi \;\Big|\; \mathtt{open}\, M_\chi \;\Big|\; \overline{\mathtt{open}}\, M_\chi$$

Table II. The reduction rules of SA

$n[\,\mathtt{in}\,m.\,P_1 \mid P_2\,] \mid m[\,\overline{\mathtt{in}}\,m.\,Q_1 \mid Q_2\,] \longrightarrow m[\,n[\,P_1 \mid P_2\,] \mid Q_1 \mid Q_2\,]$	R-In		
$m[\,n[\,\mathtt{out}\,m.\,P_1 \mid P_2\,] \mid \overline{\mathtt{out}}\,m.\,Q_1 \mid Q_2\,] \longrightarrow n[\,P_1 \mid P_2\,] \mid m[\,Q_1 \mid Q_2\,]$	R-Out		
$\mathtt{open}\,n.\,P \mid n[\,\overline{\mathtt{open}}\,n.\,Q_1 \mid Q_2\,] \longrightarrow P \mid Q_1 \mid Q_2$	R-Open		
$\langle M\rangle \mid (x : W)P \longrightarrow P\{{}^{M}\!/\!{}_{x}\}$	R-Com		
$\dfrac{P \equiv P' \qquad P' \longrightarrow P'' \qquad P'' \equiv P'''}{P \longrightarrow P'''}$	R-Struct		
$\dfrac{P \longrightarrow P'}{(\boldsymbol{\nu} n : Amb)\,P \longrightarrow (\boldsymbol{\nu} n : Amb)\,P'}$	R-Res		
$\epsilon.\,P \longrightarrow P$	R-Eps	$\mathtt{rec}\,X.\,P \longrightarrow P\{{}^{\mathtt{rec}\,X.\,P}\!/\!{}_{X}\}$	R-Rec
$\dfrac{P \longrightarrow P'}{n[\,P\,] \longrightarrow n[\,P'\,]}$	R-Amb	$\dfrac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}$	R-Par

These productions define the *colored capabilities*. Processes of CSA are the *colored processes*. A colored process P is *well-formed* if:

—the colored capabilities of P are all and only the simple capabilities in P that are unguarded and are not in the body of a recursion;

—the assignment of colors to capabilities is injective—distinct capabilities take distinct colors.

Structural congruence, $\equiv$ is defined on CSA with the same rules as for SA (Table III). In a CSA reduction $P \longrightarrow_{\mathcal{C}} P'$, the process P on the left must be well-formed and $\mathcal{C}$ is a set of colors. The rules for reduction are as in SA, except that the colors of the consumed prefixes are recorded in the reduction. The reduction axioms are shown in Table IV, where $\emptyset$ stands for the empty set.

It is straightforward to see that the refined semantics is equivalent to the original one. If P is a CSA process, then P_{erase} is the SA process obtained from P by stripping off all the colors.

Lemma 3.1. *Let $P \in$ CSA. We have:*

—if $P \longrightarrow_{\mathcal{C}} P'$, then $P_{\text{erase}} \longrightarrow P'_{\text{erase}}$.

—conversely, if $P_{\text{erase}} \longrightarrow Q$, then there are $P', \mathcal{C}$ such that $P \longrightarrow_{\mathcal{C}} P'$ and $P'_{\text{erase}} = Q$.

Intuitively, two reductions interfere if they give rise to a non-confluent diagram. Moreover, we can distinguish among plain and grave interferences: an interference

Table III. Structural congruence

$P \equiv P$	S-Refl	$\dfrac{P \equiv Q}{Q \equiv P}$	S-Symm
$\dfrac{P \equiv Q \qquad Q \equiv R}{P \equiv R}$	S-Trans	$\dfrac{P \equiv Q}{(\boldsymbol{\nu} n : Amb)\, P \equiv (\boldsymbol{\nu} n : Amb)\, Q}$	S-Res
$\dfrac{P \equiv Q}{P \mid R \equiv Q \mid R}$	S-Par	$\dfrac{P \equiv Q}{M[\,P\,] \equiv M[\,Q\,]}$	S-Amb
$\dfrac{P \equiv Q}{M.\, P \equiv M.\, Q}$	S-Pref	$\dfrac{P \equiv Q}{(x : W)P \equiv (x : W)Q}$	S-Abstr
$\dfrac{P \equiv Q}{\texttt{rec}\, X.\, P \equiv \texttt{rec}\, X.\, Q}$	S-Rec	$P \mid \mathbf{0} \equiv P$	S-ParNil
$P \mid Q \equiv Q \mid P$	S-ParCom	$P \mid (Q \mid R) \equiv (P \mid Q) \mid R$	S-ParAss
$(\boldsymbol{\nu} n : Amb)\, \mathbf{0} \equiv \mathbf{0}$	S-ResNil	$(M_1 .\, M_2).\, P \equiv M_1 .\, M_2 .\, P$	S-Path

$$(\boldsymbol{\nu} n : Amb)\,(\boldsymbol{\nu} n' : Amb')\; P \equiv (\boldsymbol{\nu} n' : Amb')\,(\boldsymbol{\nu} n : Amb)\; P \qquad \text{S-ResRes}$$

$$(\boldsymbol{\nu} n : Amb)\,(P \mid Q) \equiv ((\boldsymbol{\nu} n : Amb)\; P) \mid Q \qquad (\text{if } n \notin \mathrm{fn}(Q)) \quad \text{S-ResPar}$$

$$m[\,(\boldsymbol{\nu} n : Amb)\; P\,] \equiv (\boldsymbol{\nu} n : Amb)\; m[\,P\,] \qquad (\text{if } n \neq m) \qquad \text{S-ResAmb}$$

Table IV. The reduction axioms of CSA

$$n[\,\texttt{in}\, m_{\chi_1} .\, P_1 \mid P_2\,] \mid m[\,\overline{\texttt{in}}\, m_{\chi_2} .\, Q_1 \mid Q_2\,] \longrightarrow_{\{\chi_1, \chi_2\}} m[\,n[\,P_1 \mid P_2\,] \mid Q_1 \mid Q_2\,] \qquad \text{CR-In}$$

$$m[\,n[\,\texttt{out}\, m_{\chi_1} .\, P_1 \mid P_2\,] \mid \overline{\texttt{out}}\, m_{\chi_2} .\, Q_1 \mid Q_2\,] \longrightarrow_{\{\chi_1, \chi_2\}} n[\,P_1 \mid P_2\,] \mid m[\,Q_1 \mid Q_2\,] \qquad \text{CR-Out}$$

$$\texttt{open}\, n_{\chi_1} .\, P \mid n[\,\overline{\texttt{open}}\, n_{\chi_2} .\, Q_1 \mid Q_2\,] \longrightarrow_{\{\chi_1, \chi_2\}} P \mid Q_1 \mid Q_2 \qquad \text{CR-Open}$$

$$\langle M \rangle \mid (x : W)P \longrightarrow_{\emptyset} P\{{}^{M}\!/x\} \qquad \text{CR-Com}$$

$$\epsilon.\, P \longrightarrow_{\emptyset} P \quad \text{CR-Eps} \qquad\qquad \texttt{rec}\, X.\, P \longrightarrow_{\emptyset} P\{{}^{\texttt{rec}\, X.\, P}\!/X\} \quad \text{CR-Rec}$$

is plain if the two reductions share one of the consumed capabilities, it is grave otherwise.

Definition 3.2. **(Interferences in CSA)** Let $P \in$ CSA, and $\mathcal{C}_1, \mathcal{C}_2$ two non-empty sets of colors. Two reductions $P \longrightarrow_{\mathcal{C}_1} P_1$ and $P \longrightarrow_{\mathcal{C}_2} P_2$ *interfere on* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ if there is no P_3 such that $P_1 \longrightarrow_{\mathcal{C}_2} P_3$ and $P_2 \longrightarrow_{\mathcal{C}_1} P_3$.

P *has an interference on* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ if there are two transitions emanating from P that interfere on $\mathcal{C}_1$ and $\mathcal{C}_2$.

P *has a plain interference* if there are $\mathcal{C}_1$, $\mathcal{C}_2$ with $\mathcal{C}_1 \cap \mathcal{C}_2 \neq \emptyset$ such that P has an interference on $\mathcal{C}_1$ and $\mathcal{C}_2$.

P *has a grave interference* if there are disjoint $\mathcal{C}_1$, $\mathcal{C}_2$ such that P has an interference on $\mathcal{C}_1$ and $\mathcal{C}_2$.

Definition 3.3. **(Interferences in SA)** Let $P \in SA$.

P *has a plain interference* if there exists a well-formed colored process $P^\star$ such that $P^\star_{\text{erase}} = P$ and $P^\star$ has a plain interference.

P *has a grave interference* if there exists a well-formed colored process $P^\star$ such that $P^\star_{\text{erase}} = P$ and $P^\star$ has a grave interference.

Two simple capabilities that can be executed in parallel inside an ambient do not always produce an interference. This happens for instance when the two capabilities are coactions different from $\overline{\texttt{open}}$, as in the process

$$n[\,\overline{\texttt{in}}\,n.\,P \mid \overline{\texttt{out}}\,n.\,Q \mid m[\,\texttt{out}\,n.\,R\,]\,] \mid h[\,\texttt{in}\,n.\,S\,].$$

The 'in' and 'out' reductions can be executed in any order, and they do not affect each other. Note that if we exchange the polarity of the $\overline{\texttt{in}}$ and $\texttt{in}$ capability, as in

$$n[\,\texttt{in}\,h.\,P \mid \overline{\texttt{out}}\,n.\,Q \mid m[\,\texttt{out}\,n.\,R\,]\,] \mid h[\,\overline{\texttt{in}}\,h.\,S\,],$$

we do have a (grave) interference: the place where m will end up depends on whether $\texttt{out}\,n$ is executed before or after the movement of n.

Similarly, two parallel capabilities may not interfere when one of them is an $\texttt{open}$. For instance, in the following process the opening of m has no influence on the movement of n inside h, and conversely:

$$n[\,\texttt{in}\,h.\,P \mid \texttt{open}\,m.\,Q \mid m[\,\overline{\texttt{open}}\,m.\,R\,]\,] \mid h[\,\overline{\texttt{in}}\,h.\,S\,].$$

For Definitions 3.2 and 3.3 of interferences, the presence of coactions in reductions is essential. Therefore, as they are, the definitions cannot be used in MA. To adapt them to MA, colors should be assigned also to ambients, and the colors recorded in a movement reduction should be that of the consumed capability and that of the ambient which passively undergoes the movement. However, the distinction between plain and grave interferences would be less sharp, because certain grave interferences would have overlapping redexes. An example is given by the (colored) MA process

$$\texttt{open}\,n_{\chi_1} \mid n^{\chi}[\,\mathbf{0}\,] \mid m[\,\texttt{in}\,n_{\chi_2}.\,Q\,]$$

in which the conflict between the 'open' and 'in' movement is a grave interference. The interfering redexes share the color χ, and thus overlap. By contrast, in the corresponding (colored) SA process

$$\mathtt{open}\, n_{\chi_1} \mid n[\, \overline{\mathtt{in}}\, n_{\chi_3} \mid \overline{\mathtt{open}}\, n_{\chi_4}\,] \mid m[\, \mathtt{in}\, n_{\chi_2}.\, Q\,]$$

the redexes are decoupled, due to the presence of coactions.

We show in Appendix A a definition of plain and grave interferences as a list of all possible syntactic patterns that can produce an interference without using colors. The corresponding list of all plain and grave interferences for MA can simply be obtained removing coactions.

4. TYPES

We present three forms of types: *basic*, *immobility*, and *single-threaded* (ST) types. The basic types are an adaptation of Cardelli and Gordon's system for MA [Cardelli and Gordon 1999c], and simply ensure that there are no disagreements on the type of values communicated inside ambients. The immobility types define ambients that do not move. The ST types define ambients that, at any time, have only one possible movement. Both immobility and ST types guarantee absence of grave interferences, as we prove in Section 5. Separately, these two type systems have a limited expressiveness. However, their union gives us a powerful system – all examples we have looked at typable with basic types are also typable with immobility and ST types.

We consider the basic types to show that type systems for MA can be adapted to SA; moreover, basic types can be interesting in cases where grave interferences are not important because their typing rules are simpler.

We present the syntax and the typing rules of the three systems using several tables, with the aim of separating as much as possible the specific features of each system and of ensuring maximal orthogonality in their usage. Table V presents general productions and rules, needed in any type system. Then there are specific table for each form of type: basic, immobile, ST (Tables VI, VII, VIII, IX). Table X has additional rules for systems that have both basic and ST types; similarly Table XI, for immobility and ST types. For instance, an SA calculus with basic types only, analogous to Cardelli and Gordon's simply-typed MA [Cardelli and Gordon 1999c], is defined by Tables V and VI; whereas an SA calculus with immobility and ST types only, is defined by Tables V, VII, VIII, IX and XI. 'Union of tables' means, of course, union of the productions and of the typing rules in the tables.

In each type system, there are types of three kinds: types for ambients, for processes, and for capabilities. Metavariables *Amb*, *Proc* and *Cap* range over ambient, process and capability types, respectively. Italic characters distinguish the metavariables. A *type environment* is a finite set of assignments of types to names, variables and recursion variables. The *variables of a type environment* Γ are the set of variables on which Γ is defined.

The general grammar productions, in Table V, show that the messages exchanged in communications may be ambient or capability expressions; or there may be no exchange of messages, which is indicated by the type $\mathtt{Shh}$. Other useful type constructs can be added to the grammar of message types; for instance products, to allow the exchange of tuples of messages, or primitive constructs such as integers and booleans, or recursive types (which can be added using the standard typing

Table V. General productions and rules, needed in any type system

Grammars

			Types of expressions					Types of messages
W	$:=$	Amb	*(ambient type)*		T	$:=$	W	*(expression)*
	$\mid$	Cap	*(capability type)*			$\mid$	$\mathtt{Shh}$	*(no exchange)*

Typing of expressions

$$\frac{}{\Gamma \vdash \epsilon : Cap}\ \text{G-eps} \qquad \frac{\Gamma(n) = W}{\Gamma \vdash n : W}\ \text{G-name} \qquad \frac{\Gamma(x) = W}{\Gamma \vdash x : W}\ \text{G-varExp}$$

Typing of processes

$$\frac{\Gamma, n : Amb \vdash P : Proc}{\Gamma \vdash (\boldsymbol{\nu} n : Amb)\, P : Proc}\ \text{G-res} \qquad \frac{}{\Gamma \vdash \mathbf{0} : Proc}\ \text{G-nil}$$

$$\frac{\Gamma, X : Proc \vdash P : Proc}{\Gamma \vdash \mathtt{rec}\, X. P : Proc}\ \text{G-rec} \qquad \frac{\Gamma(X) = Proc}{\Gamma \vdash X : Proc}\ \text{G-varProc}$$

rules [Amadio and Cardelli 1993; Cardelli 1996]). We shall use some of them in our examples of Section 8.

Basic types. Basic type systems for channel-passing calculi such as the π-calculus [Sangiorgi and Walker 2001] ensure that all messages exchanged along a given name belong to the same type. Cardelli and Gordon's basic type system for MA [Cardelli and Gordon 1999c] has a similar goal and flavour, with ambient names playing the role of channel names: all messages exchanged in a given ambient should belong to the same type. It is straightforward to adapt the system to SA. The type production and type rules specific to basic types are shown in Table VI. The only type for ambients is $\mathtt{BAmb}[T]$; it is the type of an ambient inside which messages of type T may be exchanged. The only type for capabilities is $\mathtt{BCap}[T]$; this is the type of a capability that, when used, may cause the unleashing of messages of type T (this will only happen upon opening an ambient; $\mathtt{BCap}[T]$ does not record any information on the other capabilities). We call ambients of type $\mathtt{BAmb}[T]$ *basic ambients*, and capabilities of type $\mathtt{BCap}[T]$ *basic capabilities.* $\mathtt{BProc}[T]$ describes processes local to basic ambients.

A basic ambient can only open, and be opened by, other basic ambients (rules B-opencoopen) with the same type of messages. Basic ambients will be allowed to open (some of) the single-threaded ambients of the next section (see Table X). A basic ambient can however jump into an ambient of any kind (rule B-inout). Jumping into an ambient without being opened can nevertheless be useful, for instance for loading subambients and carrying them away. A coaction that types as a basic capability can only be used within a basic ambient (rule B-opencoopen and B-co-inout).

Table VI. Additional productions and rules for basic types

Grammars

			Types of expressions				*Types of processes*
Amb	$:=$	$\mathtt{BAmb}\,[T]$	*(basic ambient)*	$Proc$	$:=$	$\mathtt{BProc}\,[T]$	*(basic process)*
Cap	$:=$	$\mathtt{BCap}\,[T]$	*(basic capability)*				

Typing of expressions

$$\frac{M \in \{\mathsf{open}\,N, \overline{\mathsf{open}}\,N\} \qquad \Gamma \vdash N \ : \mathtt{BAmb}\,[T]}{\Gamma \vdash M \ : \mathtt{BCap}\,[T]} \quad \text{B-opencoopen}$$

$$\frac{M \in \{\mathtt{in}\,N, \mathtt{out}\,N\} \qquad \Gamma \vdash N \ : Amb}{\Gamma \vdash M \ : \mathtt{BCap}\,[S]} \quad \text{B-inout}$$

$$\frac{M \in \{\overline{\mathtt{in}}\,N, \overline{\mathtt{out}}\,N\} \qquad \Gamma \vdash N \ : \mathtt{BAmb}\,[T]}{\Gamma \vdash M \ : \mathtt{BCap}\,[S]} \quad \text{B-co-inout}$$

$$\frac{\Gamma \vdash M_1 \ : \mathtt{BCap}\,[T] \qquad \Gamma \vdash M_2 \ : \mathtt{BCap}\,[T]}{\Gamma \vdash M_1.\,M_2 \ : \mathtt{BCap}\,[T]} \quad \text{B-path}$$

Typing of processes

$$\frac{\Gamma \vdash M \ : \mathtt{BCap}\,[T] \qquad \Gamma \vdash P \ : \mathtt{BProc}\,[T]}{\Gamma \vdash M.\,P \ : \mathtt{BProc}\,[T]} \quad \text{B-pref}$$

$$\frac{\Gamma \vdash M \ : \mathtt{BAmb}\,[T] \qquad \Gamma \vdash P \ : \mathtt{BProc}\,[T]}{\Gamma \vdash M[\,P\,] \ : Proc} \quad \text{B-amb}$$

$$\frac{\Gamma \vdash P_1 \ : \mathtt{BProc}\,[T] \qquad \Gamma \vdash P_2 \ : \mathtt{BProc}\,[T]}{\Gamma \vdash P_1 \mid P_2 \ : \mathtt{BProc}\,[T]} \quad \text{B-par}$$

$$\frac{\Gamma \vdash M \ : W}{\Gamma \vdash \langle M \rangle \ : \mathtt{BProc}\,[W]} \quad \text{B-msg}$$

$$\frac{\Gamma, x : W \vdash P \ : \mathtt{BProc}\,[W]}{\Gamma \vdash (x : W)P \ : \mathtt{BProc}\,[W]} \quad \text{B-abs}$$

Note the difference in the type rules B-co-inout and B-inout. A coaction $\overline{\mathtt{in}}\,n$ expresses an activity that must be executed inside n[3]; hence the typing of $\overline{\mathtt{in}}\,n$ requires n itself to be a basic ambient. By contrast, $\mathtt{in}\,n$ expresses an activity that can be executed by another ambient, possibly of a different kind; hence the typing of $\mathtt{in}\,n$ does not look at the the kind of n.

Immobility types. The first refinement of the basic type system tracks ambients

[3]We recall that n refers to the ambient which is traversed.

Table VII. Additional productions and rules for immobility types

Grammars

			Types of expressions					*Types of processes*
Amb	$:=$	$\texttt{IAmb}\,[T]$	*(immobile ambients)*		$Proc$	$:=$	$\texttt{IProc}\,[T]$	*(immobile processes)*
Cap	$:=$	$\texttt{ICap}\,[T]$	*(immobile capabilities)*					

Typing of expressions

$$\frac{M \in \{\overline{\texttt{in}}\,N, \overline{\texttt{out}}\,N\} \qquad \Gamma \vdash N\ : \texttt{IAmb}\,[T]}{\Gamma \vdash M\ : \texttt{ICap}\,[S]} \quad \text{I-co-inout}$$

$$\frac{\Gamma \vdash M_1\ : \texttt{ICap}\,[T] \qquad \Gamma \vdash M_2\ : \texttt{ICap}\,[T]}{\Gamma \vdash M_1.\, M_2\ : \texttt{ICap}\,[T]} \quad \text{I-path}$$

Typing of processes

$$\frac{\Gamma \vdash M\ : \texttt{ICap}\,[T] \qquad \Gamma \vdash P\ : \texttt{IProc}\,[T]}{\Gamma \vdash M.\, P\ : \texttt{IProc}\,[T]} \quad \text{I-pref}$$

$$\frac{\Gamma \vdash M\ : \texttt{IAmb}\,[T] \qquad \Gamma \vdash P\ : \texttt{IProc}\,[T]}{\Gamma \vdash M[\,P\,]\ : Proc} \quad \text{I-amb}$$

$$\frac{\Gamma \vdash P_1\ : \texttt{IProc}\,[T] \qquad \Gamma \vdash P_2\ : \texttt{IProc}\,[T]}{\Gamma \vdash P_1 \mid P_2\ : \texttt{IProc}\,[T]} \quad \text{I-par}$$

$$\frac{\Gamma \vdash M\ : W}{\Gamma \vdash \langle M \rangle\ : \texttt{IProc}\,[W]} \quad \text{I-msg}$$

$$\frac{\Gamma, x : W \vdash P\ : \texttt{IProc}\,[W]}{\Gamma \vdash (x : W)P\ : \texttt{IProc}\,[W]} \quad \text{I-abs}$$

that: (i) cannot jump into or out of other ambients; (ii) cannot be opened. We call them *immobile ambients*. An immobile ambient can however be inside another one that jumps. For immobile ambients, we add the productions in Table VII to the type grammars. They have the following intuitive meaning:

—$\texttt{IAmb}\,[T]$ is the type of an ambient that is immobile and in whose interior messages of type T can be exchanged.

—$\texttt{ICap}\,[T]$ is the type of a capability that can be exercised inside an immobile ambient and that may unleash messages of type T.

—$\texttt{IProc}\,[T]$ is the type of a process that is local to an immobile ambient and may unleash messages of type T.

Immobile ambients have behavioural restrictions that basic ambients do not have; the typing rules of the former are indeed a subset of those for of latter. The rules for immobile ambients are shown in Table VII. With these rules, capabilities $\texttt{in}\,n$ and $\texttt{out}\,n$ cannot be exercised inside an immobile ambient. Basic ambients can jump

Table VIII. Additional productions for ST types

		Thread rights				*Types of messages*
I, J	$:= \uparrow$	*(thread)*	Tt	$:=$	$I\,W$	*(ST message)*
	$\mid\ \dagger$	*(no thread)*		$\mid$	$\mathtt{Shh}$	*(no message)*
		Types of expressions				*Types of processes*
Amb	$:= \mathtt{STAmb}\, I\,[Tt]$	*(ST ambient)*	$Proc$	$:=$	$\mathtt{STProc}\, I\,[Tt]$	*(ST process)*
Cap	$:= \mathtt{STCap}\, I\,[Tt]$	*(ST capability)*				

into an immobile ambient (rules B-inout of Table VI and I-co-inout of Table VII), but cannot be opened there. Immobile ambients will be allowed to open (some of) the single-threaded ambients of the next section (see Table XI).

Single-threadness types. The capabilities of the local processes of an ambient control the activities of that ambient. In a basic or immobile ambient such control is distributed over the local processes, in the sense that any of them may exercise a capability. In a *single-threaded* (ST) ambient, by contrast, at any moment at most *one* process has the control thread, and may therefore use a capability. An ST ambient n is willing to engage in at most one interaction at a time with external or internal ambients. Inside n, however, several activities may take place concurrently: a subambient may reduce, two subambients may interact with each other, messages may be consumed (if messages in n carry no thread, see below).

The control thread may move between processes local to an ST ambient by means of an open action. Consider, for instance, a reduction

$$n[\,\mathtt{open}\,m.\,P \mid m[\,\overline{\mathtt{open}}\,m.\,Q\,]\,] \longrightarrow n[\,P \mid Q\,] \tag{7}$$

where n and m are ST ambients. Initially $\mathtt{open}\,m.\,P$ has the control thread over n, and $\overline{\mathtt{open}}\,m.\,Q$ over m. At the end, m has disappeared; the control thread over n may or may not have moved from P to Q, depending on the type of m. If the movement occurs, Q can immediately exercise a capability, whereas P cannot; to use further capabilities within n, P will have to get the thread back.

To gain flexibility, we also allow messages carrying the thread. The type of an ambient says whether messages exchanged within it carry the thread or not. If they do, then the control thread over that ambient may move, other than by an open operation, as a result of the consumption of a message by an abstraction. This happens for instance in

$$n[\,\mathtt{in}\,h.\,\langle m\rangle \mid (x)\;\mathtt{in}\,x.\,Q\,] \mid h[\,\overline{\mathtt{in}}\,h.\,R\,] \longrightarrow\!\!\longrightarrow h[\,R \mid n[\,\mathtt{in}\,m.\,Q\,]\,] \tag{8}$$

The thread over n is first held by $\mathtt{in}\,h.\,\langle m\rangle$, but after the consumption of m the thread passes to (the derivative of) $\mathtt{in}\,x.\,Q$. In an ST ambient, when messages carry a thread, at any moment at most one capability or message are ready to fire. Therefore there can be non-determinism on reception of messages (two abstractions competing for the same message) but not on emission (two messages competing for the same abstraction).

Table IX. Additional rules for ST types

Typing of expressions

$$\frac{\Gamma \vdash M\ : \texttt{STAmb}\, I\, [Tt]}{\Gamma \vdash \texttt{open}\, M\ : \texttt{STCap}\, I^{-1}\, [Tt]} \ \text{ST-open} \qquad \frac{\Gamma \vdash M\ : \texttt{STAmb}\, I\, [Tt]}{\Gamma \vdash \overline{\texttt{open}}\, M\ : \texttt{STCap}\, I\, [Tt]} \ \text{ST-coopen}$$

$$\frac{\Gamma \vdash M_1\ : \texttt{STCap}\ \uparrow\ [Tt] \qquad \Gamma \vdash M_2\ : \texttt{STCap}\, I\, [Tt]}{\Gamma \vdash M_1.\, M_2\ : \texttt{STCap}\, I\, [Tt]} \ \text{ST-path}$$

$$\frac{M \in \{\texttt{in}\, N, \texttt{out}\, N\} \qquad \Gamma \vdash N\ : Amb}{\Gamma \vdash M\ : \texttt{STCap}\ \uparrow\ [Tt]} \ \text{ST-inout}$$

$$\frac{M \in \{\overline{\texttt{in}}\, N, \overline{\texttt{out}}\, N\} \qquad \Gamma \vdash N\ : \texttt{STAmb}\, I\, [Tt]}{\Gamma \vdash M\ : \texttt{STCap}\ \uparrow\ [Tt']} \ \text{ST-co-inout}$$

Typing of processes

$$\frac{\Gamma \vdash M\ : \texttt{STCap}\, I\, [Tt] \qquad \Gamma \vdash P\ : \texttt{STProc}\, I\, [Tt]}{\Gamma \vdash M.\, P\ : \texttt{STProc}\ \uparrow\ [Tt]} \ \text{ST-pref}$$

$$\frac{\Gamma \vdash M\ : \texttt{STAmb}\, I\, [Tt] \qquad \Gamma \vdash P\ : \texttt{STProc}\ \uparrow\ [Tt]}{\Gamma \vdash M[\, P\,]\ : Proc} \ \text{ST-amb}$$

$$\frac{\Gamma \vdash P_1\ : \texttt{STProc}\ \uparrow\ [Tt] \qquad \Gamma \vdash P_2\ : \texttt{STProc}\ \dagger\ [Tt]}{\Gamma \vdash P_1 \mid P_2\ : \texttt{STProc}\ \uparrow\ [Tt]} \ \text{ST-par1}$$

$$\frac{\Gamma \vdash P_1\ : \texttt{STProc}\ \dagger\ [Tt] \qquad \Gamma \vdash P_2\ : \texttt{STProc}\ \uparrow\ [Tt]}{\Gamma \vdash P_1 \mid P_2\ : \texttt{STProc}\ \uparrow\ [Tt]} \ \text{ST-par2}$$

$$\frac{\Gamma \vdash P_1\ : \texttt{STProc}\ \dagger\ [Tt] \qquad \Gamma \vdash P_2\ : \texttt{STProc}\ \dagger\ [Tt]}{\Gamma \vdash P_1 \mid P_2\ : \texttt{STProc}\ \dagger\ [Tt]} \ \text{ST-par3}$$

$$\frac{\Gamma \vdash M\ : W}{\Gamma \vdash \langle M \rangle\ : \texttt{STProc}\ \uparrow\ [\uparrow\ W]} \ \text{ST-msg1} \qquad \frac{\Gamma \vdash M\ : W}{\Gamma \vdash \langle M \rangle\ : \texttt{STProc}\ J\, [\dagger\, W]} \ \text{ST-msg2}$$

$$\frac{\Gamma, x : W \vdash P\ : \texttt{STProc}\, I\, [\, I\, W]}{\Gamma \vdash (x : W)P\ : \texttt{STProc}\, J\, [\, I\, W]} \ \text{ST-abs} \qquad \frac{\Gamma \vdash \texttt{rec}\, X.\, P\ : \texttt{STProc}\ \dagger\ [Tt]}{\Gamma \vdash \texttt{rec}\, X.\, P\ : \texttt{STProc}\ \uparrow\ [Tt]} \ \text{ST-rec}$$

The additional productions and rules for ST types are shown in Tables VIII and IX. We use I and J as thread tags, indicating absence ($\dagger$) or presence ($\uparrow$) of thread. The intuitive meaning of the new grammar productions is this:

—$\texttt{STAmb}\, I\, [Tt]$ is the type of an ambient that when opened: has thread right I; may unleash messages of type Tt.

—$\texttt{STCap}\, I\, [Tt]$ is the type of a capability that when used: leaves thread right I; may cause the unleash of messages of type Tt.

—Type $\mathtt{STProc}\, I\, [Tt]$ is the type of a process that has thread right I and that can unleash messages of type Tt.

—$I\, W$ is the type of a message that carries: a thread right I (that is, if $I = \upharpoonleft$ then consuming the message means acquiring the thread, whereas if $I = \dagger$ then the thread is not acquired); an expression of type W as a content.

We comment the rules for ST types. The rules for ST ambients are rather different from the rules of existing type systems for MA, and is therefore worth explaining some of them.

Rule ST-open is used when an ST ambient is opened by another ST ambient, as in (7) above. In the rule, $\upharpoonleft^{-1}$ is $\dagger$, meaning that a capability $\mathtt{open}\, n$ has the thread iff n has not, and conversely for $\dagger^{-1}$. Rule ST-coopen says that the thread rights of capability $\overline{\mathtt{open}}\, n$ are the thread rights of ambient n. Rules ST-inout and ST-co-inout show that an ST ambient can enter or exit any kind of ambient and can be traversed by any kind of ambients (basic, immobile, ST). Rule ST-amb assigns any process type to a well-typed ST ambient, which may therefore be used as a local process inside any kind of ambient. Rule ST-pref shows that a prefixed process has always the thread; its continuation keeps the thread iff the capabilities in the prefix allow so. If the prefix has the thread then it passes the thread to its continuation; if the prefix has no thread, then the continuation has no thread, which is then passed (typically by $\mathtt{open}\, n$) to a parallel process. The rules for parallel composition guarantee that at most one of the components has the thread.

For instance, in (7) if the type of Q is $\mathtt{STProc}\, \dagger\, [Tt]$ then (by rule ST-pref) $\overline{\mathtt{open}}\, m$ has type $\mathtt{STCap}\ \dagger\ [Tt]$. As a consequence (by rule ST-coopen) ambient m has type $\mathtt{STAmb}\ \dagger\ [Tt]$ showing that, when opened, m has no tread. It means that capability $\mathtt{open}\, m$ passes the thread over n to its continuation P. In fact (by rule ST-open) $\mathtt{open}\, m$ has type $\mathtt{STProc}\ \upharpoonleft\ [Tt]$ and (by rule ST-pref) $\mathtt{open}\, n.\, P$ can be typed only if P is of type $\mathtt{STProc}\ \upharpoonleft\ [Tt]$.

By contrast, if the type of Q is $\mathtt{STProc}\ \upharpoonleft\ [Tt]$ then (by rule ST-pref) $\overline{\mathtt{open}}\, m$ has type $\mathtt{STCap}\ \upharpoonleft\ [Tt]$. As a consequence (by rule ST-coopen) ambient m has type $\mathtt{STAmb}\ \upharpoonleft\ [Tt]$ showing that, when opened, m has the tread. It means that capability $\mathtt{open}\, m$ passes the thread over n to the liberated process Q. In fact (by rule ST-open) $\mathtt{open}\, m$ has type $\mathtt{STProc}\ \dagger\ [Tt]$ and (by rule ST-pref) $\mathtt{open}\, n.\, P$ can be typed only if P is of type $\mathtt{STProc}\ \dagger\ [Tt]$.

The rules for messages and abstractions have to take into account whether messages carry a thread; for instance, in ST-abs the body of the abstraction is typed as a process whose thread right is dictated by the thread right of the messages.

For instance, in (8) the type of n must be of the form $\mathtt{STAmb}\, I\, [\upharpoonleft\ W]$, to indicate that messages in n carry a thread. This allows to type the body $\mathtt{in}\, x.\, Q$ of the abstraction as a process with the thread.

Tables X and XI shows the additional rules for basic and immobility types in presence of ST types. Rules ST-B-open1, ST-B-open2, ST-I-open1 and ST-I-open2 show that ST ambients can be opened by a basic or immobile ambient; the process so liberated has no thread, nor can it acquire the thread by means of message consumption. For immobile ambients, such strict rules are necessary: subject reduction would otherwise fail. For basic ambients, the motivation is the following: in an ST ambient, a capability or a message carrying the thread are “expected” to

Table X. Additional rules rules for expressions, for systems with both basic and ST types

$\dfrac{\Gamma \vdash M \ : \texttt{STAmb}\ \dagger\ [\dagger\, W]}{\Gamma \vdash \mathsf{open}\, M \ : \texttt{BCap}\, [W]}$	ST-B-open1
$\dfrac{\Gamma \vdash M \ : \texttt{STAmb}\ \dagger\ [\texttt{Shh}]}{\Gamma \vdash \mathsf{open}\, M \ : \texttt{BCap}\, [\texttt{Shh}]}$	ST-B-open2

Table XI. Additional rules for expressions, for systems with both immobility and ST types

$\dfrac{\Gamma \vdash M \ : \texttt{STAmb}\ \dagger\ [\dagger\, W]}{\Gamma \vdash \mathsf{open}\, M \ : \texttt{ICap}\, [W]}$	ST-I-open1
$\dfrac{\Gamma \vdash M \ : \texttt{STAmb}\ \dagger\ [\texttt{Shh}]}{\Gamma \vdash \mathsf{open}\, M \ : \texttt{ICap}\, [\texttt{Shh}]}$	ST-I-open2

be exercised without interferences from the other local processes; this would not be assured if the ambient is opened by a basic ambient.

Certain parts of the type system for single-threadness—for instance, the occurrences of tag J in ST-msg2 and ST-abs, as well as generic rules such as G-nil—show that the thread is an *affine* (at most one process can hold it) rather than a *linear* (exactly one process can hold it) resource; see Lemma 5.4 of Section 5. Rule ST-rec has been added precisely to have this lemma.

Examples 4.1 and 4.2 below show: the type derivations for the typing of (7) and (8); the usefullness of ST types for the typing of immobile ambients (rules of Table XI). The typability of the process in Example 4.2 shows that a type $\texttt{STAmb}\dagger[Tt]$ for an ambient n does not prevent n from moving, since the thread right of n depends only on the thread right of the continuation of $\overline{\mathsf{open}}\, n$ (rule ST-coopen and rule ST-pref). Therefore, type $\texttt{STAmb}\ \dagger\ [Tt]$ only means that the local processes of n cannot use capabilities once $\overline{\mathsf{open}}\, n$ is consumed.

Example 4.1. Assume $\Gamma \vdash m \ : \texttt{STAmb}\ \uparrow\ [Tt]$, and $\Gamma \vdash n \ : \texttt{STAmb}\, I\, [Tt]$ with $I \in \{\uparrow, \dagger\}$, and $\Gamma \vdash P \ : \texttt{STProc}\ \dagger\ [Tt]$ and $\Gamma \vdash Q \ : \texttt{STProc}\ \uparrow\ [Tt]$. Then we have $\Gamma \vdash m[\,\overline{\mathsf{open}}\, m.\, Q\,] \ : \texttt{STProc}\ \dagger\ [Tt]$ and $\Gamma \vdash \mathsf{open}\, m.\, P \ : \texttt{STProc}\ \uparrow\ [Tt]$, as shown by the following derivations:

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash m \ : \texttt{STAmb}\ \uparrow\ [Tt]}{\Gamma \vdash \overline{\mathsf{open}}\, m \ : \texttt{STCap}\ \uparrow\ [Tt]} \qquad \Gamma \vdash Q \ : \texttt{STProc}\ \uparrow\ [Tt]}{\Gamma \vdash \overline{\mathsf{open}}\, m.\, Q \ : \texttt{STProc}\ \uparrow\ [Tt]} \qquad \Gamma \vdash m \ : \texttt{STAmb}\ \uparrow\ [Tt]}{\Gamma \vdash m[\,\overline{\mathsf{open}}\, m.\, Q\,] \ : \texttt{STProc}\ \dagger\ [Tt]}$$

$$\dfrac{\dfrac{\Gamma \vdash m \ : \texttt{STAmb}\ \uparrow\ [Tt]}{\Gamma \vdash \mathsf{open}\, m \ : \texttt{STCap}\ \dagger\ [Tt]} \qquad \Gamma \vdash P \ : \texttt{STProc}\ \dagger\ [Tt]}{\Gamma \vdash \mathsf{open}\, m.\, P \ : \texttt{STProc}\ \uparrow\ [Tt]}$$

Hence, we have

$$\dfrac{\Gamma \vdash \mathsf{open}\, m.\, P \ : \texttt{STProc}\ \uparrow\ [Tt] \qquad \Gamma \vdash m[\,\overline{\mathsf{open}}\, m.\, Q\,] \ : \texttt{STProc}\ \dagger\ [Tt]}{\Gamma \vdash \mathsf{open}\, m.\, P \mid m[\,\overline{\mathsf{open}}\, m.\, Q\,] \ : \texttt{STProc}\ \uparrow\ [Tt]}$$

and

$$\frac{\Gamma \vdash \mathtt{open}\, m.\, P \mid m[\,\overline{\mathtt{open}}\, m.\, Q\,] \ : \mathtt{STProc} \uparrow [Tt] \qquad \Gamma \vdash n \ : \mathtt{STAmb}\, I\, [Tt]}{\Gamma \vdash n[\,\mathtt{open}\, m.\, P \mid m[\,\overline{\mathtt{open}}\, m.\, Q\,]\,] \ : \mathit{Proc}}$$

Similarly, for $\Gamma \vdash m \ : \mathtt{STAmb} \dagger [Tt]$, and $\Gamma \vdash n \ : \mathtt{STAmb}\, I\, [Tt]$ with $I \in \{\uparrow, \dagger\}$, and $\Gamma \vdash P \ : \mathtt{STProc} \uparrow [Tt]$ and $\Gamma \vdash Q \ : \mathtt{STProc} \dagger [Tt]$, we have

$$\Gamma \vdash n[\,\mathtt{open}\, m.\, P \mid m[\,\overline{\mathtt{open}}\, m.\, Q\,]\,] \ : \mathit{Proc}\, .$$

Assume $\Gamma \vdash n \ : \mathtt{STAmb}\, I\, [\uparrow W]$ with $I \in \{\uparrow, \dagger\}$ and $W = \mathtt{BAmb}\, [T]$, and $\Gamma \vdash m \ : W$, and $\Gamma \vdash h \ : \mathtt{BAmb}\, [T']$, and $\Gamma \vdash R \ : \mathtt{BProc}\, [T']$, and $\Gamma, x : W \vdash Q \ : \mathtt{STProc} \uparrow [\uparrow W]$. Then we have $\Gamma \vdash \mathtt{in}\, h.\, \langle m \rangle \ : \mathtt{STProc} \uparrow [\uparrow W]$ and $\Gamma \vdash (x)\mathtt{in}\, x.\, Q \ : \mathtt{STProc} \dagger [\uparrow W]$:

$$\frac{\dfrac{\Gamma \vdash h \ : \mathtt{BAmb}\, [T']}{\Gamma \vdash \mathtt{in}\, h \ : \mathtt{STCap} \uparrow [\uparrow W]} \qquad \dfrac{\Gamma \vdash m \ : W}{\Gamma \vdash \langle m \rangle \ : \mathtt{STProc} \uparrow [\uparrow W]}}{\Gamma \vdash \mathtt{in}\, h.\, \langle m \rangle \ : \mathtt{STProc} \uparrow [\uparrow W]}$$

$$\frac{\dfrac{\dfrac{\Gamma, x : W \vdash x \ : W}{\Gamma, x : W \vdash \mathtt{in}\, x \ : \mathtt{STProc} \uparrow [\uparrow W]} \qquad \Gamma, x : W \vdash Q \ : \mathtt{STProc} \uparrow [\uparrow W]}{\Gamma, x : W \vdash \mathtt{in}\, x.\, Q \ : \mathtt{STProc} \uparrow [\uparrow W]}}{\Gamma \vdash (x)\ \mathtt{in}\, x.\, Q \ : \mathtt{STProc} \dagger [\uparrow W]}$$

Hence, we have

$$\frac{\Gamma \vdash \mathtt{in}\, h.\, \langle m \rangle \ : \mathtt{STProc} \uparrow [\uparrow W] \qquad \Gamma \vdash (x)\ \mathtt{in}\, x.\, Q \ : \mathtt{STProc} \dagger [\uparrow W]}{\Gamma \vdash \mathtt{in}\, h.\, \langle m \rangle \mid (x)\ \mathtt{in}\, x.\, Q \ : \mathtt{STProc} \uparrow [\uparrow W]}$$

and

$$\frac{\Gamma \vdash \mathtt{in}\, h.\, \langle m \rangle \mid (x)\ \mathtt{in}\, x.\, Q \ : \mathtt{STProc} \uparrow [\uparrow W] \qquad \Gamma \vdash n \ : \mathtt{STAmb}\, I\, [\uparrow W]}{\Gamma \vdash n[\,\mathtt{in}\, h.\, \langle m \rangle \mid (x)\ \mathtt{in}\, x.\, Q\,] \ : \mathit{Proc}}$$

Note that in order to type $(x)\mathtt{in}\, x.\, Q$ as $\mathtt{STProc} \dagger [\uparrow W]$ it is necessary that messages carry the thread. We have also $\Gamma \vdash h[\,\overline{\mathtt{in}}\, h.\, R\,] \ : \mathit{Proc}$:

$$\frac{\dfrac{\dfrac{\Gamma \vdash h \ : \mathtt{BAmb}\, [T']}{\Gamma \vdash \overline{\mathtt{in}}\, h \ : \mathtt{BCap}\, [T']} \qquad \Gamma \vdash R \ : \mathtt{BProc}\, [T']}{\Gamma \vdash \overline{\mathtt{in}}\, h.\, R \ : \mathtt{BProc}\, [T']} \qquad \Gamma \vdash h \ : \mathtt{BAmb}\, [T']}{\Gamma \vdash h[\,\overline{\mathtt{in}}\, h.\, R\,] \ : \mathit{Proc}}$$

Hence,

$$\frac{\Gamma \vdash n[\,\mathtt{in}\, h.\, \langle m \rangle \mid (x)\ \mathtt{in}\, x.\, Q\,] \ : \mathit{Proc} \qquad \Gamma \vdash h[\,\overline{\mathtt{in}}\, h.\, R\,] \ : \mathit{Proc}}{\Gamma \vdash n[\,\mathtt{in}\, h.\, \langle m \rangle \mid (x)\ \mathtt{in}\, x.\, Q\,] \mid h[\,\overline{\mathtt{in}}\, h.\, R\,] \ : \mathit{Proc}}$$

Similarly for $\Gamma \vdash n \ : \mathtt{STAmb}\, I\, [\uparrow W]$ with $I \in \{\uparrow, \dagger\}$ and $W = \mathtt{STAmb}\, J\, [Tt']$, and $\Gamma \vdash m \ : W$, and $\Gamma \vdash h \ : \mathtt{STAmb}\, J'\, [Tt'']$, with $J, J' \in \{\uparrow, \dagger\}$, and $\Gamma \vdash R \ : \mathtt{STProc} \uparrow [Tt'']$, and $\Gamma, x : W \vdash Q \ : \mathtt{STProc} \uparrow [Tt]$, we have

$$\Gamma \vdash n[\,\mathtt{in}\, h.\, \langle m \rangle \mid (x)\ \mathtt{in}\, x.\, Q\,] \mid h[\,\overline{\mathtt{in}}\, h.\, R\,] \ : \mathit{Proc}\, .$$

$\square$

Example 4.2. Consider the process

$$Q \stackrel{\text{def}}{=} n[\,\overline{\mathtt{in}}\,n \mid \mathtt{open}\,h \mid (x:W)P\,] \mid h[\,\mathtt{in}\,n.\,\overline{\mathtt{open}}\,h.\,\langle M\rangle\,]$$

Ambient h moves into n, where h is opened and the message $\langle M\rangle$ is emitted. Ambient h is mobile—it moves. Even though n opens a mobile ambient, n can be typed as immobile. Precisely, suppose $\Gamma \vdash M : W$, and $\Gamma \vdash h : \mathtt{STAmb}\ \dagger\ [\dagger\, W]$, and $\Gamma \vdash n : \mathtt{IAmb}\,[W]$, and $\Gamma, x : W \vdash P : \mathtt{IProc}\,[W]$; then we have

$$\Gamma \vdash Q : \mathit{Proc}\ .$$

The ST type $\mathtt{STAmb}\ \dagger\ [\dagger\, W]$ for h shows that this ambient will not keep the thread after being opened. This means that all movement capabilities ($\mathtt{in}$ and $\mathtt{out}$) inside h are consumed *before* h is opened. It is therefore safe for an immobile ambient, such as n, to open h.

In MA, without coactions and hence without ST types, types for immobility are less precise. For instance, without the coaction $\overline{\mathtt{open}}\,h$, the syntax of ambient h would not show whether h is opened before or after its jump. Indeed, the type system for controlling mobility in MA [Cardelli et al. 1999] forbids immobile ambients from opening *any* mobile ambients, precisely because the processes so liberated could, in principle, use $\mathtt{in}$ or $\mathtt{out}$ capabilities. Immobile ambients may only open other immobile ambients. Hence, in the MA version of process Q above, process n would not be typable as immobile.

We shall find similar situations in the examples of the firewall protocol and of the channel encoding in Section 8: the SA versions of the firewall agent w and the channel ambients $p^{\mathtt{ch}}$ will be typable as immobile, whereas their MA versions are not. □

5. CORRECTNESS OF THE TYPE SYSTEMS

We prove in this section some fundamental results on the correctness of the type systems. In this section, unless otherwise stated, the results are given for arbitrary processes—they may have free variables and free recursion variables. We recall that a process is closed on recursion variables if it has no free recursion variables.

Lemma 5.1. **(weakening)** *If* $\Gamma \vdash P : \mathit{Proc}$, *and the domain of* Γ *and* Γ' *are disjoint, then* $\Gamma, \Gamma' \vdash P : \mathit{Proc}$.

Lemma 5.2. **(strengthening)** *If* $\Gamma, \Gamma' \vdash P : \mathit{Proc}$ *and the names and variables in the domain of* Γ' *do not appear in* P, *then also* $\Gamma \vdash P : \mathit{Proc}$.

Lemma 5.3. **(substitution)** *If* $\Gamma, x : W \vdash P : \mathit{Proc}$ *and* $\Gamma \vdash M : W$ *then also* $\Gamma \vdash P\{^{M}\!/\!_{x}\} : \mathit{Proc}$.

Lemma 5.4. *Suppose* P *is closed on recursion variables. If* $\Gamma \vdash P : \mathtt{STProc}\,\dagger\,[Tt]$ *then also* $\Gamma \vdash P : \mathtt{STProc}\ \uparrow\ [Tt]$.

Proof. By induction on the depth of the proof of $\Gamma \vdash P : \mathtt{STProc}\ \dagger\ [Tt]$. The last rules used may have been B-amb, I-amb, G-res, G-nil, G-rec, ST-amb, ST-par3, ST-msg2, ST-abs. For rule G-rec, we need rule ST-rec. All the other cases are simple. □

Lemma 5.5. *Suppose* P *is closed on recursion variables.*

(*1*) *If* $\Gamma \vdash P\ :\texttt{STProc}\dagger\ [\dagger W]$ *(resp.* $\Gamma \vdash P\ :\texttt{STProc}\dagger\ [\texttt{Shh}]$ *) then also* $\Gamma \vdash P\ :\texttt{BProc}[W]$ *(resp.* $\Gamma \vdash P\ :\texttt{BProc}[\texttt{Shh}]$ *) .*

(*2*) *If* $\Gamma \vdash P\ :\texttt{STProc}\dagger\ [\dagger W]$ *(resp.* $\Gamma \vdash P\ :\texttt{STProc}\dagger\ [\texttt{Shh}]$ *) then also* $\Gamma \vdash P\ :\texttt{IProc}[W]$ *(resp.* $\Gamma \vdash P\ :\texttt{IProc}[\texttt{Shh}]$ *).*

Proof. As an example, we prove (2), in the case when $\Gamma \vdash P\ :\texttt{STProc}\dagger\ [\dagger W]$. The other cases are similar.

We prove a more general result, on processes that need not be closed on recursion variables. If

$$\Gamma, \tilde{X}\ :\texttt{STProc}\dagger\ [\dagger W] \vdash P\ :\texttt{STProc}\dagger\ [\dagger W]$$

where $\tilde{X}$ are all recursion variable that appear free in P, then

$$\Gamma, \tilde{X}\ :\texttt{IProc}[W] \vdash P\ :\texttt{IProc}[W]\ .$$

This is proved by induction on the depth of the proof of $\Gamma, \tilde{X}\ :\texttt{STProc}\dagger[\dagger W] \vdash P\ :\texttt{STProc}\dagger[\dagger W]$. The last rule used may have been B-amb, I-amb, G-res, G-nil, G-rec, G-varProc, ST-amb, ST-par3, ST-msg2, ST-abs. All cases are simple. For instance, consider ST-abs: the premise must be $\Gamma, x : W \vdash P\ :\texttt{STProc}\dagger\ [\dagger W]$; by induction $\Gamma, x\ :\ W \vdash P\ :\texttt{IProc}[W]$; hence, by rule I-abs, $\Gamma \vdash (x\ :\ W)P\ :\texttt{IProc}[W]$. In the case of B-amb, I-amb, and ST-amb, the assertion follows immediately using strengthening and weakening, because recursion variables cannot occur free inside an ambient. □

Lemmas 5.4 and 5.5 show that the type systems allow a (limited) form of subtyping.

Remark 5.6. **(recursion variables)** Lemma 5.5 would fail without the constraint on recursive definitions that the recursion variable should not appear free underneath an ambient. As a counterexample, take process $\texttt{rec}\,X.\,m[\,\overline{\texttt{in}}\,m\ |\ X\,]$ where m is an ST ambient; this process could be assigned a type $\texttt{STProc}\dagger\ [Tt]$, but not an immobility or basic type (precisely because the recursion variable appears inside an ST ambient).

There are other reasons why the present rules are not enough to handle recursive definitions in which the recursion variable appears free underneath an ambient. See for instance the discussion after Corollary 5.9. □

Lemma 5.7. *If* $\Gamma \vdash P\ : Proc$ *and* $P \equiv Q$ *then also* $\Gamma \vdash Q\ : Proc$ *.*

Proof. Suppose $\Gamma \vdash P\ : Proc$. One then proves that $P \equiv Q$ or $Q \equiv P$ implies $\Gamma \vdash Q\ : Proc$, proceeding by induction on the depth of the proof of $P \equiv Q$. All cases are simple. In the case of rule S-ParNil, Lemma 5.4 is needed. □

Lemma 5.7 would fail if rule R-eps were a structural rule, as in MA [Cardelli and Gordon 1998a]. For instance, $\mathbf{0}$ could be typed as $\texttt{STProc}\dagger\ [Tt]$, for any Tt, but $\epsilon.\,\mathbf{0}$ could not.

Lemma 5.8. $\Gamma, X : Proc \vdash P\ : Proc$ *implies* $\Gamma \vdash P\{^{\texttt{rec}\,X.\,P}/X\}\ : Proc$ *.*

Proof. If $\Gamma, X\ : Proc \vdash P\ : Proc$, then by rule G-rec, also

$$\Gamma \vdash \texttt{rec}\,X.\,P\ : Proc\ . \tag{9}$$

Consider the derivation proof tree of $\Gamma, X : \mathit{Proc} \vdash P : \mathit{Proc}$. This tree can be modified so to obtain a proof tree for $\Gamma \vdash P\{^{\mathtt{rec}\,X.\,P}/_X\} : \mathit{Proc}$; the main manipulation is the following. Consider the nodes in the tree of $\Gamma, X : \mathit{Proc} \vdash P : \mathit{Proc}$ in which the term being typed is X; that is, the judgement in the node is of the form $\Delta \vdash X : \mathit{Proc}$, where Δ is an extension of Γ. Such a judgement must have been inferred using rule G-varProc. This means that the node is a leaf and, using weakening, it may be replaced by the proof tree of (9). $\square$

Corollary 5.9. *$\Gamma \vdash \mathtt{rec}\,X.P : \mathit{Proc}$, where $\mathtt{rec}\,X.P$ is closed on recursion variables, implies $\Gamma \vdash P\{^{\mathtt{rec}\,X.\,P}/_X\} : \mathit{Proc}$.*

Proof. The last rules used in the proof of $\Gamma \vdash \mathtt{rec}\,X.P : \mathit{Proc}$ is either G-rec or ST-rec. In the former case, the premise is $\Gamma, X : \mathit{Proc} \vdash P : \mathit{Proc}$, and therefore $\Gamma \vdash P\{^{\mathtt{rec}\,X.\,P}/_X\} : \mathit{Proc}$ can be inferred from Lemma 5.8.

The latter case is only possible if Proc is of the form $\mathtt{STProc} \uparrow [\mathit{Tt}]$. In this case the premise of the rule is $\Gamma \vdash \mathtt{rec}\,X.P : \mathtt{STProc} \dagger [\mathit{Tt}]$. This must have been derived using rule G-rec; we can therefore apply the result of the corollary and infer $\Gamma \vdash P\{^{\mathtt{rec}\,X.\,P}/_X\} : \mathtt{STProc} \dagger [\mathit{Tt}]$. By Lemma 5.4, also $\Gamma \vdash P\{^{\mathtt{rec}\,X.\,P}/_X\} : \mathtt{STProc} \uparrow [\mathit{Tt}]$. $\square$

Probably also the converse of Corollary 5.9 holds, but we shall not need it. However the converse would fail without the constraint on recursive definitions that the recursion variable should not appear free underneath an ambient. As a counterexample, take process $\mathtt{rec}\,X.m[\,\overline{\mathtt{in}}\,h \mid \overline{\mathtt{in}}\,k.X\,]$, where m is an immobile ambient. It would be possible to type

$$m[\,\overline{\mathtt{in}}\,h \mid \overline{\mathtt{in}}\,k.\,X\,]\{^{\mathtt{rec}\,X.\,m[\overline{\mathtt{in}}\,h \,\mid\, \overline{\mathtt{in}}\,k.\,X\,]}/_X\}$$

with an ST type, but not $\mathtt{rec}\,X.m[\,\overline{\mathtt{in}}\,h \mid \overline{\mathtt{in}}\,k.X\,]$. We have

$$\Gamma \vdash m[\,\overline{\mathtt{in}}\,h \mid \overline{\mathtt{in}}\,k.\,X\,]\{^{\mathtt{rec}\,X.\,m[\overline{\mathtt{in}}\,h \,\mid\, \overline{\mathtt{in}}\,k.\,X\,]}/_X\} : I\;\mathtt{STProc}\,I\,[\mathit{Tt}]$$

in two cases only: m is basic or m is immobile. Ambient m cannot be typed ST, regadless of the type of the recursive process, because there are two capabilities in parallel inside m. Suppose m is basic, say $\Gamma(m) = \mathtt{BAmb}[T]$. To obtain $\Gamma \vdash \mathtt{rec}\,X.m[\,\overline{\mathtt{in}}\,h \mid \overline{\mathtt{in}}\,k.X\,] : \mathtt{STProc}\,I\,[\mathit{Tt}]$ we should have (by rule G-rec) $\Gamma, X : \mathtt{STProc}\,I\,[\mathit{Tt}] \vdash m[\,\overline{\mathtt{in}}\,h \mid \overline{\mathtt{in}}\,k.X\,] : \mathtt{STProc}\,I\,[\mathit{Tt}]$, which is not possible because m is basic and cannot have local ST processes. Similarly if m were immobile.

The three theorems below assert the soundness of the typing rules. The first theorem shows preservation of types under reduction. The other two theorems show that immobile and ST processes do respect the behavioural restrictions that distinguish the corresponding types, as informally described in Section 4.

Theorem 5.10. *If $\Gamma \vdash P : \mathit{Proc}$, with P closed, and $P \longrightarrow P'$ then also $\Gamma \vdash P' : \mathit{Proc}$.*

Proof. By induction on the depth of the proof of $P \longrightarrow P'$. The hardest cases are the axioms. We consider them in the sequel.

R-In We have $P = n[\,\mathtt{in}\,m.P_1 \mid P_2\,] \mid m[\,\overline{\mathtt{in}}\,m.Q_1 \mid Q_2\,]$, and $P' = m[\,n[\,P_1 \mid P_2\,] \mid Q_1 \mid Q_2\,]$. Processes P and P' can be given any type Proc provided that the ambients n and m are well-typed. This because the rules for typing ambients

(B-amb, I-amb, and ST-amb) allows us to assign any type to a well-typed ambient. We proceed by considering all possible combinations of types for n and m: basic, ST, and immobile. The case when n is immobile is impossible, because there is no typing rule for immobile ambients with which process $\mathtt{in}\, m.\, P_1$ could be typed. We only show the details for when n is basic and m is ST; the remaining cases are similar.

Thus the type of m is of the form $\mathtt{STAmb}\, I\, [Tt]$. By rule ST-amb, process $\overline{\mathtt{in}}\, m.\, Q_1 \mid Q_2$ must have type $\mathtt{STProc} \uparrow [Tt]$. By rules ST-co-inout and ST-pref, the type of Q_1 is $\mathtt{STProc} \uparrow [Tt]$; hence, because of the rules of ST for parallel composition, the type of Q_2 must be $\mathtt{STProc} \dagger [Tt]$.

On the other hand, the type of n is of the form $\mathtt{BAmb}\, [T]$, and hence, by rule B-amb that of $\mathtt{in}\, m.\, P_1 \mid P_2$ must be $\mathtt{BProc}\, [T]$; hence, by the rules for prefixing and parallel composition of basic processes, processes P_i have type $\mathtt{BProc}\, [T]$ too.

Reversing the reasoning, we can thus prove that P' has the same type as P. Process $P_1 \mid P_2$ has type $\mathtt{BProc}\, [T]$; by rule B-amb we can infer that $n[\, P_1 \mid P_2\,]$ has type $\mathtt{STProc} \dagger [Tt]$. Thus, using the types above for Q_1 and Q_2, we can infer that $n[\, P_1 \mid P_2\,] \mid Q_1 \mid Q_2$ has type $\mathtt{STProc} \uparrow [Tt]$. Hence the ambient m of P' is well-typed and by rule ST-amb it can be assigned the type *Proc*.

R-Out Similar to R-In.

R-Open We have $P = \mathtt{open}\, n.\, P_1 \mid n[\, \overline{\mathtt{open}}\, n.\, Q_1 \mid Q_2\,]$ and $P' = P_1 \mid Q_1 \mid Q_2$. We have to distinguish all possible combinations of types for *Proc* and $\Gamma(n)$.

(1) *Proc* is basic and $\Gamma(n)$ is ST. In this case the capability $\mathtt{open}\, n$ has been typed using either rule ST-B-open1 or ST-B-open2. Therefore either $\Gamma \vdash n : \mathtt{STAmb} \dagger [\dagger W]$ and *Proc* is $\mathtt{BProc}\, [W]$, or $\Gamma \vdash n : \mathtt{STAmb} \dagger [\mathtt{Shh}]$ and *Proc* is $\mathtt{BProc}\, [\mathtt{Shh}]$. Suppose the former is true (the case when the latter is true is similar). Using rule B-pref we infer that the type of P_1 is $\mathtt{BProc}\, [W]$, and using the rules for ST types (such as ST-coopen) we infer that the type of Q_1 and Q_2 is $\mathtt{STProc} \dagger [\dagger W]$. By Lemma 5.5 (1), Q_1 and Q_2 may also be given type $\mathtt{BProc}\, [W]$. Hence also process $P_1 \mid Q_1 \mid Q_2$ may be given type $\mathtt{BProc}\, [W]$.

(2) *Proc* is immobile and $\Gamma(n)$ is ST. This is similar to the previous case.

(3) *Proc* is ST and $\Gamma(n)$ is basic. This case is impossible: the capability $\mathtt{open}\, n$ could not be typed (an ST process may only open ST ambients).

(4) *Proc* is immobile and $\Gamma(n)$ is basic. This case is impossible: the capability $\mathtt{open}\, n$ could not be typed (an immobile process may only open ST ambients).

(5) Both *Proc* and $\Gamma(n)$ are ST. There are two subcases, depending on whether $\Gamma(n)$ is of the form $\mathtt{STAmb} \uparrow [Tt]$, or $\mathtt{STAmb} \dagger [Tt]$. We consider the former, the latter is similar. Because of rule ST-pref, it holds that $\Gamma \vdash P : \mathtt{STProc} \uparrow [Tt]$. We have to show that P' can be typed in the same way. By rule ST-open1 and ST-pref we infer $\Gamma \vdash P_1 : \mathtt{STProc} \dagger [Tt]$; on the other hand, using ST-coopen and other ST rules, we infer $\Gamma \vdash Q_1 : \mathtt{STProc} \uparrow [Tt]$ and $\Gamma \vdash Q_2 : \mathtt{STProc} \dagger [Tt]$. Thus, using the rules for parallel composition, $\Gamma \vdash P' : \mathtt{STProc} \uparrow [Tt]$.

(6) Both *Proc* and $\Gamma(n)$ are basic. This is conceptually similar to, but much simpler than, the previous case.

R-Com We have $P = \langle M \rangle \mid (x : W)Q$ and $P' = Q\{M/x\}$. We proceed by case analysis on the type *Proc*.

(1) *Proc* is of the form $\mathtt{STProc} \dagger [\dagger W]$. We have $\Gamma \vdash M : W$ and $\Gamma, x : W \vdash Q : \mathtt{STProc} \dagger [\dagger W]$. By the substitution Lemma 5.3, also $\Gamma \vdash Q\{M/x\} : \mathtt{STProc} \dagger [\dagger W]$.

(2) *Proc* is of the form $\mathtt{STProc} \uparrow [\dagger W]$. We have $\Gamma \vdash M : W$ and $\Gamma, x : W \vdash Q : \mathtt{STProc} \dagger [\dagger W]$. By the substitution Lemma 5.3, also $\Gamma \vdash Q\{M/x\} : \mathtt{STProc} \dagger [\dagger W]$. By Lemma 5.4, $\Gamma \vdash Q\{M/x\} : \mathtt{STProc} \uparrow [\dagger W]$.

(3) *Proc* is of the form $\mathtt{STProc} \dagger [\uparrow W]$. This case is not possible (see rules ST-msg1 and ST-msg2).

(4) *Proc* is of the form $\mathtt{STProc} \uparrow [\uparrow W]$. We have $\Gamma \vdash M : W$ and $\Gamma, x : W \vdash Q : \mathtt{STProc} \uparrow [\uparrow W]$. By the substitution Lemma 5.3, also $\Gamma \vdash Q\{M/x\} : \mathtt{STProc} \uparrow [\uparrow W]$.

(5) *Proc* is a basic or immobility type. Immediate from the substitution lemma.

R-Rec This is Corollary 5.9.

R-Eps In this case P is $\epsilon. Q$ and P' is Q. Regardless of whether *Proc* is basic, ST, or immobile, in the typing of P also process Q is typed as *Proc*.

For the inference rules, we only show the proof for R-Struct:

$$\frac{P \equiv P' \qquad P' \longrightarrow P'' \qquad P'' \equiv P'''}{P \longrightarrow P'''}$$

If $\Gamma \vdash P : Proc$, then by Lemma 5.7 also $\Gamma \vdash P' : Proc$; by induction, $\Gamma \vdash P'' : Proc$; again by Lemma 5.7, $\Gamma \vdash P''' : Proc$. □

We say that an occurrence of an expression M in a process P is *unguarded* if that occurrence is not underneath a prefix, an ambient, or an abstraction; likewise for unguarded messages.

THEOREM 5.11. **(immobility types)** *If* $\Gamma \vdash P : \mathtt{IProc}[T]$ *then* P *has no unguarded capabilities of the form* $\mathtt{in}\,h$, $\mathtt{out}\,h$ *or* $\overline{\mathtt{open}}\,h$.

PROOF. By induction on the depth of the proof of $\Gamma \vdash P : \mathtt{IProc}[T]$. The last rules used may have been I-pref, B-amb, I-amb, ST-amb, I-par, G-res, I-abs, I-msg, G-nil, G-rec, G-varProc. Cases ST-amb, B-amb, , I-amb, I-abs, I-msg, G-nil are obvious. In case of I-pref we have $P = M.Q$ with $\Gamma \vdash M : \mathtt{IProc}[T]$ and $\Gamma \vdash Q : \mathtt{IProc}[T]$. By the typing rules for expressions, an expression M cannot be typed $\mathtt{IProc}[T]$, with $M \in \{\mathtt{in}\,h, \mathtt{out}\,h, \overline{\mathtt{open}}\,h\}$ for some h. □

THEOREM 5.12. **(single-threadness types)**

(1) If $\Gamma \vdash P : \mathtt{STProc} \dagger [Tt]$ *then* P *has no unguarded capabilities; moreover if* $Tt = \uparrow W$ *(for some* W*) then* P *has also no unguarded message.*

(2) If $\Gamma \vdash P : \mathtt{STProc} \uparrow [Tt]$ *then* P *has at most one unguarded capability; moreover if* $Tt = \uparrow W$ *(for some* W*) then* P *has at most one unguarded capability or message.*

PROOF. (1) By induction on the proof of $\Gamma \vdash P : \mathtt{STProc} \dagger [Tt]$. The last rules used may have been ST-amb, B-amb, I-amb, ST-par3, G-res, ST-abs, ST-msg2, G-nil, G-rec, G-varProc. Cases ST-amb, B-amb, I-amb, ST-abs, ST-msg2, G-nil and G-varProc are obvious. The other cases G-rec, ST-par3 and G-res follow by induction. If $Tt =\uparrow W$ (for some W) then P has no unguarded message since rule ST-msg2 could not have been applied.

(2) By induction on the proof of $\Gamma \vdash P : \mathtt{STProc} \uparrow [Tt]$. The last rules used may have been ST-pref, ST-amb, B-amb, I-amb, ST-par1, ST-par2, G-res, ST-abs, ST-msg1, ST-msg2, G-nil, G-rec, G-varProc. Cases ST-amb, , B-amb, I-amb, ST-abs, ST-msg1, ST-msg2, G-nil and G-varProc are obvious. Cases ST-par1 and ST-par2 follow by induction and by part (1) of this theorem. If ST-pref has been applied then $P = M.Q$ with $\Gamma \vdash M : \mathtt{STCap}\, I\, [Tt]$ and $\Gamma \vdash Q : \mathtt{STProc}\, I\, [Tt]$; in this case P contains exactly one unguarded capability. For ST-rec, use part (1) of this theorem. For the remaining cases use induction. The assertion regarding $Tt =\uparrow W$ is proved in a similar way; in this case, however, rule ST-msg2 could not have been the last rule used.

□

THEOREM 5.13. *If* $\Gamma \vdash P : \mathtt{STProc}\, I\, [\mathtt{Shh}]$ *, or* $\Gamma \vdash P : \mathtt{BProc}\, [\mathtt{Shh}]$ *, or* $\Gamma \vdash P : \mathtt{IProc}\, [\mathtt{Shh}]$ *, then* P *has no unguarded message.*

PROOF. We notice that process $\langle M \rangle$ cannot be typed $\mathtt{STProc}\, I\, [\mathtt{Shh}]$ since rules ST-msg1 and ST-msg2 cannot be applied; process $\langle M \rangle$ can be typed $\mathtt{BProc}\, [\mathtt{Shh}]$ by rule B-msg, and $\mathtt{IProc}\, [\mathtt{Shh}]$ by rule I-msg, if $\Gamma \vdash M : \mathtt{Shh}$. By the typing rules for expressions no expression M exists such that $\Gamma \vdash M : \mathtt{Shh}$. □

As a consequence of Theorems 5.11 and 5.12, a system in which each ambient is either immobile or ST presents no grave interferences.

COROLLARY 5.14. **(no grave interferences)** *Suppose* $\Gamma \vdash P : Proc$ *, and* P *is closed. Suppose also that the type of each name defined in* Γ *and each restricted name in* P *is immobile or ST. For all* P' *such that* $P \Longrightarrow P'$*, process* P' *has no grave interferences.*

The proof of Corollary 5.14 is shown in the Appendix A.

The converse of Corollary 5.14 does not hold. For instance the processes below cannot be typed in type environments in which all names are ST or immobile. The first process has only a plain interference, the second process and all its derivatives have no interferences.

$$n[\,\mathtt{in}\, m \mid \mathtt{in}\, m\,] \mid m[\,\overline{\mathtt{in}}\, m\,]$$
$$n[\,\mathtt{in}\, h.\, (\overline{\mathtt{in}}\, n \mid \overline{\mathtt{out}}\, n) \mid m[\,\mathtt{out}\, n\,]\,] \mid h[\,\overline{\mathtt{in}}\, h \mid k[\,\mathtt{in}\, n\,]\,]$$

6. BEHAVIOURAL EQUIVALENCE

The behavioural equality we adopt in the paper is *barbed congruence* [Milner and Sangiorgi 1992]. Barbed congruence is a bisimulation-based relation that has been used for a broad variety of calculi; its main advantage is that it can be uniformly defined on different calculi, for it requires of the calculus little more than a notion of reduction.

Barbed congruence is defined as the congruence induced by *barbed bisimulation*. We write $P \Downarrow_n$ if P is *observable at* n; this means, intuitively, that P contains an ambient n that may accept interactions with the external environment; formally

$P \Downarrow_n$ is true if $P \Longrightarrow P'$, for some P' such that $P' \equiv \boldsymbol{\nu}\tilde{n}\,(n[\,\mu.\,Q_1 \mid Q_2\,] \mid Q_3)$ where $\mu \in \{\overline{\mathtt{in}}\,n, \overline{\mathtt{open}}\,n\}$ and $n \notin \tilde{n}$.

A *context* is a process expression with a single occurrence of a hole $[\cdot]$ in it. A context C is *closing* for a set of variables $\tilde{x}$, or a type environment Γ whose variables are $\tilde{x}$, if $C[P]$ is closed, for all P with $\mathrm{fv}(P) \subseteq \tilde{x}$. For simplicity, we often simply say that a context C is closing, without mentioning the set of variables or the type environment. We define barbed congruence on processes that are closed on recursion variables. Therefore in the remainder of the paper, all the results are on processes that are closed on recursion variables.

Definition 6.1. **(barbed bisimulation and congruence)** *Barbed bisimulation* is the largest symmetric relation $\dot{\simeq}$ on closed processes such that $P \dot{\simeq} Q$ implies:

(1) whenever $P \Longrightarrow P'$ then there exists Q' such that $Q \Longrightarrow Q'$ and $P' \dot{\simeq} Q'$;
(2) for each name n, $P \Downarrow_n$ iff $Q \Downarrow_n$.

Two processes P and Q are *barbed congruent*, written $P \simeq Q$, if $C[P] \dot{\simeq} C[Q]$ for all closing contexts C.

In an untyped calculus, no constraint is made on processes or contexts. In a typed setting, processes compared should have the same type, and should only be tested in contexts that respect their typing. A context C is a (Γ/Proc)*-context* if C is closing on the variables of Γ and $\Gamma \vdash C : \mathit{Proc}'$, for some Proc', assuming that the hole is a process of type Proc.

Definition 6.2. Two processes P and Q are *barbed congruent at Proc with respect to* Γ, written $\Gamma \triangleright P \simeq Q : \mathit{Proc}$, if both $\Gamma \vdash P : \mathit{Proc}$ and $\Gamma \vdash Q : \mathit{Proc}$, and $C[P] \dot{\simeq} C[Q]$ for all (Γ/Proc)-context C.

In the algebraic laws of Section 7, we shall control plain interferences among processes by constraining the use of capabilities in processes and in contexts. We will write laws of the form

$$\Gamma\{L\} \triangleright P_1 \simeq P_2 : \mathit{Proc}$$

where L is a set of simple capabilities. This means that $\Gamma \vdash P_i : \mathit{Proc}$ and that processes P_1 and P_2 cannot be distinguished by contexts that, besides respecting the type information Γ and Proc, never use the capabilities in L. We define a syntactic condition which ensures this requirement under the assumption that the process placed in the hole does not trasmit the capabilities of L.

Definition 6.3. Let μ be a simple capability. We say that a context C *does not know* μ if

(1) μ does not appear in prefixes or messages of C;
(2) Name $\mathrm{n}(\mu)$ is not transmitted in messages (i.e., there is no messages $\langle n \rangle$ in C, where n is the name of μ).

Similarly, we say that a process P *does not know* μ (just use P in place of C above).

If a context does not know μ, then μ cannot appear in the context, even after some reductions; condition (2) is needed because a recipient of a message n, where n is the name of μ, could reconstruct and use μ.

Definition 6.4. Let L be a non-empty set of simple capabilities. A context C respects the linearity *of* L if

(1) C does not know the capabilities in L;
(2) the hole of C does not occur under recursion.

Condition (2) guarantees that if a process possessing a capability in L is placed in the hole then this capability will not be replicated.

When contexts are supposed to respect the linearity of a set of capabilities, the notion of observation has to be modified accordingly, for the context—and hence also an observer—cannot use these capabilities. For a set of simple capabilities L, we write $P \Downarrow_n^L$ if $P \Longrightarrow P'$, for some P' such that $P' \equiv \boldsymbol{\nu}\tilde{n}\,(n[\,\mu.\,Q_1 \mid Q_2\,] \mid Q_3)$ and either $\mu = \overline{\mathtt{in}}\,n$, $n \notin \tilde{n}$, and $\mathtt{in}\,n \notin L$, or $\mu = \overline{\mathtt{open}}\,n$, $n \notin \tilde{n}$, and $\mathtt{open}\,n \notin L$. We first give the definition for the untyped calculus.

Definition 6.5. Let L be a set of simple capabilities. *Barbed bisimulation with respect to* L is the largest symmetric relation $\mathcal{R}$ on closed processes such that $P \;\mathcal{R}\; Q$ implies:

(1) whenever $P \Longrightarrow P'$ then there exists Q' such that $Q \Longrightarrow Q'$ and $P' \;\mathcal{R}\; Q'$;
(2) for each name n, $P \Downarrow_n^L$ iff $Q \Downarrow_n^L$.

We write $\{L\} \triangleright P \stackrel{\centerdot}{\simeq} Q$ if P and Q are barbed bisimilar with respect to L.

Two processes P and Q are *barbed congruent with respect to* L, written $\{L\} \triangleright P \simeq Q$, if $\{L\} \triangleright C[P] \stackrel{\centerdot}{\simeq} C[Q]$, for all closing contexts C that respect the linearity of L.

The definition for the typed case is modified accordingly: two processes P and Q are *barbed congruent at Proc with respect to* Γ *and* L, written $\Gamma\{L\} \triangleright P \simeq Q \,:\, \mathit{Proc}$, if $\Gamma \vdash P \,:\, \mathit{Proc}$, $\Gamma \vdash Q \,:\, \mathit{Proc}$ and $\{L\} \triangleright C[P] \stackrel{\centerdot}{\simeq} C[Q]$, for all (Γ/Proc)-context C that respects the linearity of L.

In $\Gamma \triangleright P \simeq Q \,:\, \mathit{Proc}$ and $\Gamma\{L\} \triangleright P \simeq Q \,:\, \mathit{Proc}$, when possible we omit *Proc* and simply write $\Gamma\{L\} \triangleright P \simeq Q$ and $\Gamma \triangleright P \simeq Q$.

7. ALGEBRAIC LAWS

Key algebraic laws for both sequential and concurrent calculi are those that make a reduction step into a behavioural equality. If the reduction relation is confluent, then all reduction steps are also behavioural equalities. But if reduction is not confluent, as usual in concurrency, then some conditions have to be imposed to obtain equalities. This section presents a sample of laws of this kind, for both untyped and typed SA. Therefore in all the laws in this section the left-hand side process may reduce in one step to the right-hand side. (Other algebraic laws are of course those for structural equivalence.)

The proofs of the laws are reported in Appendix C. Such proofs exploit the labeled transition semantics of SA presented in Appendix B.

Untyped calculus. The laws of Lemma 7.1 are already valid in untyped SA, hence also in its typed versions.

LEMMA 7.1. *We have:*

(*1*) $\boldsymbol{\nu} n\,(m[\,\mathtt{in}\,n.\,P\,] \mid n[\,\overline{\mathtt{in}}\,n.\,Q\,]) \simeq \boldsymbol{\nu} n\,(n[\,Q \mid m[\,P\,]\,])$;
(*2*) $k[\,m[\,\mathtt{in}\,n.\,P\,] \mid n[\,\overline{\mathtt{in}}\,n.\,Q\,] \mid \overline{\mathtt{out}}\,k.\,R\,] \simeq k[\,n[\,Q \mid m[\,P\,]\,] \mid \overline{\mathtt{out}}\,k.\,R\,]$;
(*3*) $\boldsymbol{\nu} m\,(\mathtt{open}\,m.\,P \mid m[\,\overline{\mathtt{open}}\,m.\,Q\,]) \simeq \boldsymbol{\nu} m\,(P \mid Q)$;
(*4*) $k[\,\mathtt{open}\,m.\,P \mid m[\,\overline{\mathtt{open}}\,m.\,Q\,]\,] \simeq k[\,P \mid Q\,]$;
(*5*) $n[\,\overline{\mathtt{out}}\,n.\,P \mid m[\,\mathtt{out}\,n.\,Q\,]\,] \simeq m[\,Q\,] \mid n[\,P\,]$;
(*6*) $n[\,\langle M\rangle \mid (x)\,P\,] \simeq n[\,P\{^{M}\!/\!_{x}\}\,]$.
(*7*) $k[\,P\,] \simeq \mathbf{0}$, *if* P *has no unguarded capability or message.*

In these equivalences, grave interferences do not arise because ambients have a single local process; plain interferences are avoided either by a restriction, or by the enclosing ambient.

In the second law, process $\overline{\mathtt{out}}\,k.\,R$ is an example of a further parallel components that is possible to have inside k; it is not the only example, but it is one that we shall use in Section 8.

Typed calculi. The laws of Lemma 7.1 are sometimes useful (for instance, they are almost enough for the algebraic proof of the firewall crossing protocol and the second encoding of channels in Section 8). In general, however, they are too restrictive, in that they are only applicable to ambients with a single local process. We obtain more powerful laws exploiting ST and immobility types. The reason why these types are useful is that they remove all forms of grave interferences (Corollary 5.14). We therefore only need conditions for controlling plain interferences. For this, we use syntactic conditions on the appearance of certain capabilities in the context and in the processes, as formalised in Section 6.

The first set of laws is about interactions between ST ambients. We write $P \stackrel{\mathtt{out}\,n}{\not\Longrightarrow}$ if no ambient in P demands to exit n; formally, there are no $m, \tilde{n}, Q_1, Q_2, Q_3$ such that $P \Longrightarrow\equiv \boldsymbol{\nu}\tilde{n}\,(m[\,\mathtt{out}\,n.\,Q_1 \mid Q_2\,] \mid Q_3)$ with $n \notin \tilde{n}$ ($P \stackrel{\mathtt{out}\,n}{\not\Longrightarrow}$ is undecidable, but one may think of various syntactic conditions that will imply it, for instance P being an abstraction or not containing n free).

LEMMA 7.2. *Suppose ambients* n *and* m *are ST in* Γ. *We have*

(*1*) $\Gamma\{\mathtt{in}\,n, \overline{\mathtt{in}}\,n\} \rhd m[\,\mathtt{in}\,n.\,P_1 \mid P_2\,] \mid n[\,\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2\,] \simeq n[\,Q_1 \mid Q_2 \mid m[\,P_1 \mid P_2\,]\,]$;
(*2*) $\Gamma\{\mathtt{open}\,m, \overline{\mathtt{open}}\,m\} \rhd \mathtt{open}\,m.\,P \mid m[\,\overline{\mathtt{open}}\,m.\,Q_1 \mid Q_2\,] \simeq P \mid Q_1 \mid Q_2$;
(*3*) $\Gamma \rhd n[\,\overline{\mathtt{out}}\,n.\,P_1 \mid m[\,\mathtt{out}\,n.\,Q_1 \mid Q_2\,] \mid P_2\,] \simeq m[\,Q_1 \mid Q_2\,] \mid n[\,P_1 \mid P_2\,]$, *if* $P_2 \stackrel{\mathtt{out}\,n}{\not\Longrightarrow}$;
(*4*) $\Gamma \rhd n[\,\langle M\rangle \mid (x)\,P \mid Q\,] \simeq m[\,P\{^{M}\!/\!_{x}\} \mid Q\,]$, *if messages in* n *carry the thread (i.e.,* $\Gamma(n)$ *is of the form* $\mathtt{STAmb}\,I\,[\uparrow W]$*) and no abstractions appear unguarded in* Q.

Law (1) does not hold without the capability constraints on $\mathtt{in}\,n$, $\overline{\mathtt{in}}\,n$, for the surrounding contexts may contain an ambient that wishes to enter n (thus interfering with m) or another ambient n that allows to be entered (thus interfering with the existing n). Similarly, the constraints on $\mathtt{open}\,m, \overline{\mathtt{open}}\,m$ are necessary in law (2).

Remark 7.3. In the laws of Lemma 7.2, types (in particular the ST type condition on ambients n and m) can be removed; the price is that the equalities of the laws have to be syntactically strengthened by eliminating processes P_2 or Q_2. Thus, if the ST type condition on ambient n is removed then P_2 should be $\mathbf{0}$, and similarly for m and Q_2. If in laws (1) and (2) the ST type condition on both n and m is removed, then the resulting laws are variants of the laws of Lemma 7.1. □

The capability constraints in laws (1) and (2) may be removed by means of restrictions or ambients:

COROLLARY 7.4. *Suppose ambients n and m are ST in Γ. We have*

(1) $\Gamma \triangleright \boldsymbol{\nu} n \left(m[\,\mathtt{in}\, n.\, P_1 \mid P_2\,] \mid n[\,\overline{\mathtt{in}}\, n.\, Q_1 \mid Q_2\,]\right) \simeq \boldsymbol{\nu} n \left(n[\,Q_1 \mid Q_2 \mid m[\,P_1 \mid P_2\,]\,]\right)$*;*

(2a) $\Gamma \triangleright \boldsymbol{\nu} m\, (\mathtt{open}\, m.\, P \mid m[\,\overline{\mathtt{open}}\, m.\, Q_1 \mid Q_2\,]) \simeq \boldsymbol{\nu} m\, (P \mid Q_1 \mid Q_2)$*;*

(2b) $\Gamma \triangleright k[\,\mathtt{open}\, m.\, P \mid m[\,\overline{\mathtt{open}}\, m.\, Q_1 \mid Q_2\,]\,] \simeq k[\,P \mid Q_1 \mid Q_2\,]$*.*

Next we formulate laws concerning interactions between ST and immobile ambients. Laws similar to those of Lemma 7.2 and Corollary 7.4 hold. We only present some laws that are specific to immobile ambients and which involve *uniformly receptive* capabilities [Sangiorgi 1997] . Roughly, a capability is uniform receptive if the unique process which may use it is of the form $!\mu.\, P$. Assuming that process $!\mu.\, P'$ does not move (which is the case here where the process is local to an immobile ambient), this condition ensures that processes attempting to consume μ do not interfere among them: μ stays in the same place, is always available, and always activates the same continuation P'.

An immobile ambient n often uses capabilities such as $\overline{\mathtt{in}}\, n$, $\overline{\mathtt{out}}\, n$, or $\mathtt{open}\, m$ in a uniformly receptive way. This happens if n continuously enables movements into and out of it, or is continuously ready to open ambients named m (examples are the channel ambients in the first encoding of channels, Section 8).

LEMMA 7.5. *Suppose that ambient n is immobile in Γ, and ambient m is ST.*

(*1*) *Assume that P_i and Q_i ($i = 1, 2$) do not know $\overline{\mathtt{in}}\, n$. Then*

$$\begin{aligned}\Gamma\{\overline{\mathtt{in}}\, n\} \triangleright \quad & m[\,\mathtt{in}\, n.\, P_1 \mid P_2\,] \mid n[\,!\overline{\mathtt{in}}\, n.\, Q_1 \mid Q_2\,] \\ & \simeq n[\,!\overline{\mathtt{in}}\, n.\, Q_1 \mid Q_1 \mid Q_2 \mid m[\,P_1 \mid P_2\,]\,].\end{aligned}$$

(*2*) *Assume that P_i and Q_i ($i = 1, 2$) do not know $\overline{\mathtt{out}}\, n$. Then*

$$\begin{aligned}\Gamma\{\overline{\mathtt{out}}\, n\} \triangleright \quad & n[\,!\overline{\mathtt{out}}\, n.\, P_1 \mid m[\,\mathtt{out}\, n.\, Q_1 \mid Q_2\,] \mid P_2\,] \\ & \simeq m[\,Q_1 \mid Q_2\,] \mid n[\,!\overline{\mathtt{out}}\, n.\, P_1 \mid P_1 \mid P_2\,].\end{aligned}$$

(*3*) *Assume that P_i and Q_i ($i = 1, 2$) do not know* $\mathtt{open}\, m$*. Then*

$$\begin{aligned}\Gamma\{\mathtt{open}\, m\} \triangleright \quad & n[\,!\mathtt{open}\, m.\, P_1 \mid m[\,\overline{\mathtt{open}}\, m.\, Q_1 \mid Q_2\,] \mid P_2\,] \\ & \simeq n[\,!\mathtt{open}\, m.\, P_1 \mid P_1 \mid Q_1 \mid Q_2 \mid P_2\,].\end{aligned}$$

In law (1), the restriction on the use of $\overline{\mathtt{in}}\, n$ both in the context and in P_i, Q_i guarantees the uniform receptiveness of $\overline{\mathtt{in}}\, n$. Similarly for laws (2) and (3).

In Lemma 7.2 and 7.5, we have used syntactic conditions on the use of certain capabilities in order to remove possible plain interferences. In the algebraic theory of name-passing process calculi such as the π-calculus, where all interferences are plain interferences, similar conditions are formalised by means of type systems; see for instance the type systems for capabilities control [Pierce and Sangiorgi 1996; Kobayashi et al. 1996; Sewell 1998], and the type systems for receptiveness [Sangiorgi 1997]. Types could probably be used also in our case, in place of the syntactic conditions; this would give more powerful laws, but would complicate the proofs of correctness.

8. EXAMPLES

There is an obvious encoding of MA into SA, where each ambient n is given unlimited availability of the coactions $\overline{\mathtt{in}}\, n$, $\overline{\mathtt{out}}\, n$, $\overline{\mathtt{open}}\, n$ (that is, process $!\overline{\mathtt{in}}\, n \mid !\overline{\mathtt{out}}\, n \mid !\overline{\mathtt{open}}\, n$ is placed inside n). There is, however, a natural and more conscientious way of rewriting MA programs into SA, whereby coactions are inserted only when actually needed, rather than "by saturation". The firewall protocol and the first encoding of channels below are examples of this. We also look at the protocols for renaming and for routing packets, from [Cardelli and Gordon 1998a]. In MA they suffer interference problems, which undermine their correctness. In SA we are able to remove most of the interferences, using coactions and types. Finally we present a different encoding of channels that has various desirable properties over the previous one. This encoding only makes sense in SA: removing coactions we obtain useless MA programs. In all examples: the ambients presented are typable with the type systems of Section 4, in some cases using also product types and recursive types; the behavioural equivalences are proved algebraically, from the laws of structural equivalence and of Section 7.

Renaming.
Here is the MA construct for changing the name of an ambient, from [Cardelli and Gordon 1998a]:

$$n\ \mathtt{be}\ m.\, P \stackrel{\mathrm{def}}{=} m[\, \mathtt{out}\, n.\, \mathtt{open}\, n.\, P\,] \mid \mathtt{in}\, m.$$

A process $n\ \mathtt{be}\ m.\, P$ placed inside an ambient n may cause the renaming of the ambient n to m thus: $n[\, n\ \mathtt{be}\ m.\, P \mid Q\,] \Longrightarrow m[\, P \mid Q\,]$. For the correctness of the protocol, we look at the conditions under which $n[\, n\ \mathtt{be}\ m.\, P \mid Q\,]$ and $m[\, P \mid Q\,]$ are behaviourally the same. This is stronger than requiring successful completion of the protocol implementing the renaming, but the correctness conditions are essentially the same and the statement is simpler.

The equality may fail in MA because, for instance: (i) the external environment contains ambients named n or m; (ii) the external environment contains ambients that try to enter n, or to open n or m; (iii) Q causes n to jump somewhere else; (iv) Q contains an ambient that tries to exit n. There are various reasons why an ambient entering n may disrupt the protocol; for instance the ambient entering could be $h[\, \mathtt{in}\, n.\, \mathtt{in}\, k\,]$ and Q could be $\mathtt{open}\, h$, in which case n may jump into k. Similarly, an ambient exiting n could interfere: if the ambient that exits is $h[\, \mathtt{out}\, n.\, \mathtt{open}\, n\,]$, then this ambient, once out of n, can be opened and then can open n (note that this interference arises even if n is unknown to the external environment).

Conditions (i-iv) are strong: in particular, (i) and (ii) essentially mean that *both n and m* are unknown to the external environment, which probably makes the renaming construct not very useful. Renaming becomes, in SA:

$$n\ \mathtt{be}\ m.\,P \stackrel{\text{def}}{=} m[\,\mathtt{out}\,n.\,\overline{\mathtt{in}}\,m.\,\mathtt{open}\,n.\,P\,] \mid \overline{\mathtt{out}}\,n.\,\mathtt{in}\,m.\,\overline{\mathtt{open}}\,n.$$

Process $n[\,n\ \mathtt{be}\ m.\,P \mid Q\,]$ is well typed in type environments in which n is ST. For instance, it holds that $\Gamma \vdash n[\,n\ \mathtt{be}\ m.\,P \mid Q\,] : \mathit{Proc}$ if Γ is such that

—$\Gamma \vdash m : \mathtt{STAmb}\,I\,[Tt]$ (for $I \in \{\uparrow, \dagger\}$),

—$\Gamma \vdash n : \mathtt{STAmb}\ \dagger\ [Tt]$,

—$\Gamma \vdash P : \mathtt{STProc}\ \uparrow\ [Tt]$,

—$\Gamma \vdash Q : \mathtt{STProc}\ \dagger\ [Tt]$.

Ambient m can also be typed basic, because n is an ST ambient without thread and basic ambients can open ST ambients without thread (rules ST-B-open1 and ST-B-open2). It may be the type of P that dictates whether m should be typed as ST or basic. In the algebraic manipulations below, we assume that m is ST in Γ; when m is basic, the variants of laws (1) and (3) of Lemma 7.2 explained in Remark 7.3 should be used.

Behaviourally, we have: if n is ST in Γ and $Q \overset{\mathtt{out}\,n}{\not\Longrightarrow}$, then

$$\Gamma\{\overline{\mathtt{in}}\,m, \mathtt{in}\,m\} \triangleright n[\,n\ \mathtt{be}\ m.\,P \mid Q\,] \simeq m[\,P \mid Q\,]. \tag{10}$$

The single threadedness of n guarantees that Q does not cause unexpected movements of n; the capability constraints on $\mathtt{in}\,m, \overline{\mathtt{in}}\,m$ guarantee that no processes in the environment interfere with the movement of ambient n into m; the semantic condition on Q guarantees that no ambients in Q try to exit n, stealing the coaction $\overline{\mathtt{out}}\,n$ to the ambient m. An ambient in the external environment trying to enter n, or to open n or m, does not interfere, since the complementary coaction is not available.

The proof of (10) is purely algebraic: since $Q \overset{\mathtt{out}\,n}{\not\Longrightarrow}$, then by law (3) of Lemma 7.2,

$$\begin{aligned}\Gamma \triangleright\quad & n[\,m[\,\mathtt{out}\,n.\,\overline{\mathtt{in}}\,m.\,\mathtt{open}\,n.\,P\,] \mid \overline{\mathtt{out}}\,n.\,\mathtt{in}\,m.\,\overline{\mathtt{open}}\,n \mid Q\,]\\ &\simeq m[\,\overline{\mathtt{in}}\,m.\,\mathtt{open}\,n.\,P\,] \mid n[\,\mathtt{in}\,m.\,\overline{\mathtt{open}}\,n \mid Q\,].\end{aligned}$$

By law (1) of Lemma 7.2,

$$\begin{aligned}\Gamma\{\overline{\mathtt{in}}\,m, \mathtt{in}\,m\} \triangleright\quad & m[\,\overline{\mathtt{in}}\,m.\,\mathtt{open}\,n.\,P\,] \mid n[\,\mathtt{in}\,m.\,\overline{\mathtt{open}}\,n \mid Q\,]\\ &\simeq m[\,n[\,\overline{\mathtt{open}}\,n \mid Q\,] \mid \mathtt{open}\,n.\,P\,].\end{aligned}$$

By law ($2b$) of Corollary 7.4,

$$\Gamma \triangleright m[\,n[\,\overline{\mathtt{open}}\,n \mid Q\,] \mid \mathtt{open}\,n.\,P\,] \simeq m[\,Q \mid P\,].$$

If the *target* name m is only known to P, then the capability constraints on $\overline{\mathtt{in}}\,m$ and $\mathtt{in}\,m$ are not necessary: if n is ST and $Q \overset{\mathtt{out}\,n}{\not\Longrightarrow}$, then we have

$$\Gamma \triangleright n[\,\boldsymbol{\nu} m\ (n\ \mathtt{be}\ m.\,P) \mid Q\,] \simeq \boldsymbol{\nu} m\ (m[\,P \mid Q\,]). \tag{11}$$

In SA we can define a more robust protocol of renaming, which works under milder conditions. Define

$$n\ \mathtt{be}^+\ m.\,P \stackrel{\text{def}}{=} \boldsymbol{\nu} h\ (\ h[\,\mathtt{out}\,n.\,\overline{\mathtt{in}}\,h.\,\overline{\mathtt{out}}\,h \mid m[\,\overline{\mathtt{in}}\,m.\,\mathtt{open}\,n\,]\,]$$
$$\mid \overline{\mathtt{out}}\,n.\,\mathtt{in}\,h.\,\mathtt{in}\,m.\,\overline{\mathtt{open}}\,n.\,\mathtt{out}\,h.\,P)$$

where h is any fresh name. The typing of process $n\ \mathtt{be}^+\ m.\,P$ is similar to that of $n\ \mathtt{be}\ m.\,P$, using a type $\mathtt{STAmb}\,I\,[Tt']$, for $I \in \{\uparrow, \dagger\}$ and Tt', for the fresh name h. In the case of $n\ \mathtt{be}^+\ m.\,P$, correctness holds without capability constraints: if n is ST in Γ, then

$$\Gamma \triangleright n[\,n\ \mathtt{be}^+\ m.\,P \mid Q\,] \simeq m[\,P \mid Q\,]. \tag{12}$$

The proofs of (11) and (12) are similar to those of (10). The main difference is that the constraints on $\mathtt{in}\,m, \overline{\mathtt{in}}\,m$ are not needed. In place of law (1) of Lemma 7.2 we use, in the proof of (11), law (1) of Corollary 7.4 and, in the proof of (12), the variant of law (2) of Lemma 7.1 explained in Remark 7.3, where an ambient with a single local process is replaced by an ST ambient;

Routable Packets.

In this protocol, from [Cardelli and Gordon 1998a], a packet can be repeatedly routed to various destinations. The content P and the destination M are contained in an ambient *route*. The act of sending P to destination M is realized by the following steps. Ambient *route* enters inside the packet and is opened. This liberates a message $\langle M\rangle$, which is then consumed so that the path M can be executed. At the end, the packet, which contains P, has reached destination. Here is the MA program:

$$PKT \stackrel{\text{def}}{=} pkt[\,!(x)\,x \mid !\mathtt{open}\,route\,] \qquad \text{(the packet)}$$
$$\langle P, M\rangle \stackrel{\text{def}}{=} route[\,\mathtt{in}\,pkt.\,\langle M\rangle \mid P\,] \qquad \text{(P is routed to destination M)}$$

This protocol, however, only works under severe constraints on process P and on the environment. Possible dangers are

(i) process P can interfere on the path to follow;
(ii) two routers could enter *pkt* and interfere on the path to follow;
(iii) *pkt* and *route* could be opened by the environment.

The first problem can be solved by properly enclosing P inside ambients (see below). The second problem is more complex—and more dangerous. To see an interference between two routers, consider the system

$$SYS \stackrel{\text{def}}{=} PKT \mid \langle P_1, \mathtt{in}\,n.\,\mathtt{in}\,k\rangle \mid \langle P_2, \mathtt{in}\,m\rangle \mid n[\,Q_1 \mid m[\,Q_2\,] \mid k[\,Q_3\,]\,] \mid m[\,Q_2\,].$$

The system has a reduction in which none of the contents P_1, P_2 reaches the destination:

$$\begin{array}{lll} SYS \Longrightarrow & n[& Q_1 \\ & & \mid m[\,Q_2 \mid pkt[\ \ !(x)\,x \mid !\mathtt{open}\,route \\ & & \qquad\qquad \mid P_1 \mid \langle \mathtt{in}\,k\rangle \mid P_2\,]\ \ ] \\ & & \mid k[\,Q_3\,]\ \ ] \\ & \mid m[\,Q_2\,]. & \end{array}$$

The packet has ended up in the inner ambient m, rather than, as intended, in the ambient k or in the outer ambient m.

Further complications would arise if we wished a router content P to exit pkt upon arrival at destination (P should not exit before arrival, but must exit before pkt moves again.) To avoid all these problems the protocol for both the packet and the route ambients has to be substantially complicated, along the lines of the taxi protocol in [Cardelli and Gordon 1999c]. Below is the SA protocol, with the additional demand on exit for P (which explains why P is placed inside an ambient $dest$).

$$\begin{aligned} PKT &\stackrel{\text{def}}{=} pkt[\,R\,] \\ \langle P, M\rangle &\stackrel{\text{def}}{=} route[\,\mathtt{in}\,pkt.\,\overline{\mathtt{open}}\,route.\,\langle M\rangle \mid dest[\,\mathtt{out}\,pkt.\,P\,]\,] \\ R &\stackrel{\text{def}}{=} \mathtt{rec}\,X.\,\overline{\mathtt{in}}\,pkt.\,\mathtt{open}\,route.\,(x)\,x.\,\overline{\mathtt{out}}\,pkt.\,X \end{aligned}$$

Ambients pkt and $route$ can be typed as ST ambients, using also recursive types. We derive $\Gamma \vdash PKT \mid \langle P, M\rangle : Proc$ if Γ is a type environment such that

—$\Gamma \vdash pkt : \mathtt{STAmb}\,I\,[Tt]$ for $I \in \{\uparrow, \dagger\}$,

—$\Gamma \vdash route : \mathtt{STAmb} \uparrow [Tt]$,

—$\Gamma \vdash M : W$,

—$\Gamma \vdash dest[\,\mathtt{out}\,pkt.\,P\,] : \mathtt{STProc} \dagger [Tt]$ (this holds if $\Gamma \vdash \mathtt{out}\,pkt.\,P : Proc$ and $\Gamma(dest)$ is defined)

where $Tt = \uparrow W$ and $W = \mathtt{STCap} \uparrow [Tt]$. The definitions of W and Tt are mutually recursive, which shows that we need recursive types to make sense of it (in type systems for Ambients or related calculi, recursive types are necessary to type a process $(x)\,x.\,P$, if messages can be capabilities).
The (grave) interferences (i)-(iii) discussed at the beginning of the example disappear:

(1) the interferences of P are controlled by ambient $dest$ in conjunction with $\overline{\mathtt{out}}\,pkt$ (in MA two levels of ambients are needed to lock P);

(2) two routers cannot enter pkt at the same time since entering is controlled by $\overline{\mathtt{in}}\,pkt$;

(3) ambients pkt and $route$ cannot be opened by the environment since opening is controlled by $\overline{\mathtt{open}}\,pkt$ and $\overline{\mathtt{open}}\,route$.

Consider the SA program corresponding to SYS, where Q_a, for $a \in \{n, m, k\}$, is of the form $!\overline{\mathtt{in}}\,a.\,Q'_a \mid Q''_a$, so that arbitrary many processes can enter ambient a:

$$PKT \mid \langle P_1, \mathtt{in}\,n.\,\mathtt{in}\,k\rangle \mid \langle P_2, \mathtt{in}\,m\rangle \mid n[\,Q_n \mid m[\,Q_m\,] \mid k[\,Q_k\,]\,] \mid m[\,Q_m\,].$$

It can make two reductions, depending on whether the first or the second router enters pkt. Suppose it is the first; the derivative is then

$$\begin{aligned} SYS_1 \stackrel{\text{def}}{=}\;\; & pkt[\;\mathtt{open}\,route.\,(x)\,x.\,\overline{\mathtt{out}}\,pkt.\,R \\ & \qquad \mid route[\,\overline{\mathtt{open}}\,route.\,\langle \mathtt{in}\,n.\,\mathtt{in}\,k\rangle \mid dest[\,\mathtt{out}\,pkt.\,P_1\,]\,]\,] \\ & \mid SYS'_1 \end{aligned}$$

where

$$SYS_1' \stackrel{\mathrm{def}}{=} \langle P_2, \mathtt{in}\, m\rangle \mid n[\, Q_n \mid m[\, Q_m\,] \mid k[\, Q_k\,]\,] \mid m[\, Q_m\,].$$

Let Γ be a type environment in which ambients pkt and $route$ are ST and n, m, k are immobile. We have $\Gamma \vdash SYS_1 : Proc$, if Γ is defined as above (where P_1 and P_2 have the typing of P) and

—$\Gamma \vdash a : \mathtt{IAmb}[T]$ for $a \in \{n, m, k\}$,
—$\Gamma \vdash Q_a : \mathtt{IProc}[T]$ for $a \in \{n, m, k\}$.

Assume also that $\overline{\mathtt{in}}\, n, \overline{\mathtt{in}}\, k$ are uniformly receptive; that is, neither Q_m and Q_k (respectively Q_n), nor Q_n' and Q_n'' (respectively Q_k' and Q_k'')) know $\overline{\mathtt{in}}\, n$ (respectively $\overline{\mathtt{in}}\, k$). Then we have

$$\begin{array}{l}\Gamma\{\overline{\mathtt{in}}\, n, \overline{\mathtt{in}}\, k\} \triangleright \\ \qquad SYS_1 \\ \quad \simeq \langle P_2, \mathtt{in}\, m\rangle \mid n[\, Q_n' \mid Q_n \mid m[\, Q_m\,] \mid k[\, Q_k' \mid Q_k \mid PKT \mid dest[\, P_1\,]\,]\,] \mid m[\, Q_m\,].\end{array}$$

The equivalence shows that P_1 will always correctly reach destination. The proof of the equivalence is completely algebraic, as we now outline.
By law (2b) of Corollary 7.4,

$$\begin{array}{ll}\Gamma \triangleright & pkt[\ \ \mathtt{open}\, route.\, (x)\, x.\, \overline{\mathtt{out}}\, pkt.\, R \\ & \qquad \mid route[\, \overline{\mathtt{open}}\, route.\, \langle \mathtt{in}\, n.\, \mathtt{in}\, k\rangle \mid dest[\, \mathtt{out}\, pkt.\, P_1\,]\,]\,] \\ & \simeq\ pkt[\, (x)\, x.\, \overline{\mathtt{out}}\, pkt.\, R \mid \langle \mathtt{in}\, n.\, \mathtt{in}\, k\rangle \mid dest[\, \mathtt{out}\, pkt.\, P_1\,]\,].\end{array}$$

By law (4) of Lemma 7.2,

$$\begin{array}{ll}\Gamma \triangleright & pkt[\, (x)\, x.\, \overline{\mathtt{out}}\, pkt.\, R \mid \langle \mathtt{in}\, n.\, \mathtt{in}\, k\rangle \mid dest[\, \mathtt{out}\, pkt.\, P_1\,]\,] \\ & \simeq pkt[\, \mathtt{in}\, n.\, \mathtt{in}\, k.\, \overline{\mathtt{out}}\, pkt.\, R \mid dest[\, \mathtt{out}\, pkt.\, P_1\,]\,].\end{array}$$

By law (1) of Lemma 7.5 and by law (5) of Lemma 7.1, we conclude

$$\begin{array}{l}\Gamma\{\overline{\mathtt{in}}\, n, \overline{\mathtt{in}}\, k\} \triangleright \\ \qquad SYS_1 \\ \quad \simeq \langle P_2, \mathtt{in}\, m\rangle \mid n[\, Q_n' \mid Q_n \mid m[\, Q_m\,] \mid k[\, Q_k' \mid Q_k \mid PKT \mid dest[\, P_1\,]\,]\,] \mid m[\, Q_m\,].\end{array}$$

The case when the second router enters is more difficult, and for this we need laws concerning interactions between ST and immobile ambients more powerful than those in Lemma 7.5. In particular, the syntactic condition on the use of capability $\overline{\mathtt{in}}\, m$ in the context should be relaxed. We leave their formalisation for future work.

Crossing a firewall.
A protocol is discussed in [Cardelli and Gordon 1998a] for controlling accesses through a firewall. Ambient w represents the firewall; ambient m represents a trusted agent inside which is a process Q that is supposed to cross the firewall. The firewall ambient sends into the agent a pilot ambient k with the capability

$\mathtt{in}\,w$ for entering the firewall. The agent acquires the capability by opening k. The process Q carried by the agent is finally liberated inside the firewall by the opening of ambients m and k'. Names m, k, k' act like passwords which guarantee the access only to authorized agents. Here is the protocol, in MA:

$$
\begin{aligned}
AG &\stackrel{\text{def}}{=} m[\,\mathtt{open}\,k.\,k'[\,Q\,]\,] \\
FW &\stackrel{\text{def}}{=} \boldsymbol{\nu} w\,\Big(w[\,\mathtt{open}\,m.\,\mathtt{open}\,k'.\,P \mid k[\,\mathtt{out}\,w.\,\mathtt{in}\,m.\,\mathtt{in}\,w\,]\,]\Big)
\end{aligned}
$$

The correctness of the protocol is shown in [Cardelli and Gordon 1999b], proving that

$$(\boldsymbol{\nu} m, k, k'\,)(FW \mid AG) \simeq \boldsymbol{\nu} w\,(w[\,Q \mid P\,])$$

under the conditions that $w \notin \mathrm{fn}(Q)$ and $(\mathrm{fn}(Q) \cup \mathrm{fn}(P)) \cap \{m, k, k'\} = \emptyset$.

The restrictions on both m and k are necessary to avoid grave interferences. For instance, the protocol would be damaged if external processes opened ambients m and k, or if external ambients entered inside m or k. The ambient k', and the condition $\{k', k, m\} \cap \mathrm{fn}(Q) = \emptyset$, are needed to block process Q, so to avoid interferences due to actions from Q. Without ambient k', process Q would be local to m, and could cause m to move somewhere else rather than into the firewall. Without the condition $\{m\} \cap \mathrm{fn}(Q) = \emptyset$, process Q could cause ambient $k'[\,Q\,]$ to jump out of m before m has entered the firewall. Similarly for the other conditions. The restriction on k' and the condition $(\mathrm{fn}(P)) \cap \{m, k, k'\} = \emptyset$, are not necessary for correctness but simplify the proofs.

Below is essentially the same protocol, written in SA.

$$
\begin{aligned}
AG &\stackrel{\text{def}}{=} m[\,\overline{\mathtt{in}}\,m.\,\mathtt{open}\,k.\,(x)\,x.\,\overline{\mathtt{open}}\,m.\,Q\,] \\
FW &\stackrel{\text{def}}{=} \boldsymbol{\nu} w\,\Big(w[\,\overline{\mathtt{out}}\,w.\,\overline{\mathtt{in}}\,w.\,\mathtt{open}\,m.\,P \mid k[\,\mathtt{out}\,w.\,\mathtt{in}\,m.\,\overline{\mathtt{open}}\,k.\,\langle \mathtt{in}\,w\rangle\,]\,]\Big)
\end{aligned}
$$

Coactions aside, there are two differences with the MA protocol: (i) ambient k' is not needed to control the movements of Q; (ii) the capability $\mathtt{in}\,w$ for entering the firewall is communicated to the agent as a message. The simplification (i) is possible because Q is blocked underneath the coaction $\overline{\mathtt{open}}\,m$. The modification (ii) is not essential, but is necessary for proving correctness with the algebraic laws of Section 7.

If $w \notin \mathrm{fn}(Q)$ and $m \notin \mathrm{fn}(P)$, we have:

$$\boldsymbol{\nu} m\,(AG \mid FW) \simeq (\boldsymbol{\nu} m, w\,)(w[\,P \mid Q\,]). \tag{13}$$

The restriction on m can be replaced by a capability constraint on $\mathtt{in}\,m$ and $\overline{\mathtt{in}}\,m$:

$$\{\mathtt{in}\,m, \overline{\mathtt{in}}\,m\} \triangleright AG \mid FW \simeq \boldsymbol{\nu} w\,(w[\,P \mid Q\,]).$$

Note that in both SA equivalences no restriction on ambient k is necessary, since opening k and entering k is controlled by coactions $\overline{\mathtt{open}}\,k$ and $\overline{\mathtt{in}}\,k$. The two equivalences can be proved purely algebraically (rather than by reasoning over contexts as in [Cardelli and Gordon 1999b]), using the untyped laws of Lemma 7.1, and the laws with capability constraints on $\mathtt{in}\,m$ and $\overline{\mathtt{in}}\,m$ indicated in Remark 7.3. Therefore, as the password k to access the firewall is not restricted, the same name can be shared by different authorized agents.

Equality (13) is obtained by combining the following transformation steps. By law (5) of Lemma 7.1,

$$w[\,\overline{\mathtt{out}}\,w.\,\overline{\mathtt{in}}\,w.\,\mathtt{open}\,m.\,P \mid k[\,\mathtt{out}\,w.\,\mathtt{in}\,m.\,\overline{\mathtt{open}}\,k.\,\langle \mathtt{in}\,w\rangle\,]\,]$$

$$\simeq\ w[\,\overline{\mathtt{in}}\,w.\,\mathtt{open}\,m.\,P\,] \mid k[\,\mathtt{in}\,m.\,\overline{\mathtt{open}}\,k.\,\langle \mathtt{in}\,w\rangle\,].$$

Using a variant of law (1) of Lemma 7.1,

$$\boldsymbol{\nu} m\, \Big(k[\,\mathtt{in}\,m.\,\overline{\mathtt{open}}\,k.\,\langle \mathtt{in}\,w\rangle\,] \mid m[\,\overline{\mathtt{in}}\,m.\,\mathtt{open}\,k.\,(x)\,x.\,\overline{\mathtt{open}}\,m.\,Q\,] \mid w[\,\overline{\mathtt{in}}\,w.\,\mathtt{open}\,m.\,P\,]\Big)$$

$$\simeq\quad \boldsymbol{\nu} m\, \Big(m[\,k[\,\overline{\mathtt{open}}\,k.\,\langle \mathtt{in}\,w\rangle\,] \mid \mathtt{open}\,k.\,(x)\,x.\,\overline{\mathtt{open}}\,m.\,Q\,] \mid w[\,\overline{\mathtt{in}}\,w.\,\mathtt{open}\,m.\,P\,]\Big)\,.$$

By laws (4) and (6) of Lemma 7.1,

$$m[\,k[\,\overline{\mathtt{open}}\,k.\,\langle \mathtt{in}\,w\rangle\,] \mid \mathtt{open}\,k.\,(x)\,x.\,\overline{\mathtt{open}}\,m.\,Q\,] \simeq m[\,\mathtt{in}\,w.\,\overline{\mathtt{open}}\,m.\,Q\,]\,.$$

Finally, by laws (1) and (4) of Lemma 7.1,

$$\boldsymbol{\nu} w\,(w[\,\overline{\mathtt{in}}\,w.\,\mathtt{open}\,m.\,P\,] \mid m[\,\mathtt{in}\,w.\,\overline{\mathtt{open}}\,m.\,Q\,]) \simeq \boldsymbol{\nu} w\,(w[\,P \mid Q\,])\,.$$

Although types are not needed for the correctness proof, it is worth pointing out that the processes of the protocol may be typed as ST, under appropriate assumptions on the unknown processes P and Q, and using also recursive types. Let Γ be a type environment such that

—$\Gamma \vdash P\ : \mathtt{STProc}\ \uparrow\ [Tt]\,,$
—$\Gamma \vdash Q\ : \mathtt{STProc}\ \dagger\ [Tt]\,,$
—$\Gamma \vdash m\ : \mathtt{STAmb}\ \dagger\ [Tt]\,,$
—$\Gamma \vdash k\ : \mathtt{STAmb}\ \uparrow\ [Tt]\,,$
—$\Gamma \vdash w\ : \mathtt{STAmb}\,I\,[Tt]$

where $Tt =\uparrow W$ and $W = \mathtt{STCap}\ \uparrow\ [Tt]$. We then have

$$\Gamma \vdash FW \mid AG\ : Proc\,.$$

We obtain the same result if the types of P and Q are exchanged. Other typings are possible, for instance one in which w is typed as immobile.

The firewall protocol in MA is also analised by Nielson et al. [Nielson et al. 1999]. The authors employ control-flow analysis techniques to compute the set of ambients that may be at the top-level of a given ambient. Using such an analysis, they prove the property that agents not knowing the appropriate passwords cannot enter the firewall. A similar analysis can be carried out in the SA formulation, as shown in [Degano et al. 2000], where the analysis is also refined by introducing a sort of contextual information.

Encoding of channels. We present the standard encoding of the polyadic asynchronous π-calculus into MA [Cardelli and Gordon 1999c; Cardelli et al. 1999], rewritten in SA (also the target SA calculus is polyadic, that is with the tuple type $W_1 \times \ldots \times W_n$ among the type for messages). The encoding is an important test of expressiveness. It also shows how to program communications along channels (in MA and in SA communications are anonymous).

First, a briefly recall of the polyadic asynchronous π-calculus. The type grammar is

$$T := Ch(T_1 \ldots T_n) \qquad n \geq 0$$

The type judgements are of the form $\Gamma \vdash P$, where Γ is a finite assignment of types to channels, and asserts that process P respects the type information in Γ. A π-calculus channels p of type $Ch(T_1 \ldots T_n)$ carries n-uples of channels, whose i-the component has type T_i. The typing rules are straightforward:

$$\frac{\Gamma \vdash p : Ch(T_1 \ldots T_n) \qquad \Gamma, q_1 : T_1 \ldots q_n : T_n \vdash P}{\Gamma \vdash p(q_1 : T_1 \ldots q_n : T_n).\, P}$$

$$\frac{\Gamma \vdash p : Ch(T_1 \ldots T_n) \qquad \Gamma \vdash q_i : T_i \quad (1 \leq i \leq n)}{\Gamma \vdash \overline{p}\langle q_1 \ldots q_n \rangle}$$

$$\frac{\Gamma \vdash P_1 \qquad \Gamma \vdash P_1}{\Gamma \vdash P_1 \mid P_2} \qquad\qquad \frac{\Gamma \vdash P}{\Gamma \vdash !P}$$

$$\frac{\Gamma, p : T \vdash P}{\Gamma \vdash (\boldsymbol{\nu} p : T)\, P} \qquad\qquad \frac{}{\Gamma \vdash \mathbf{0}}$$

In the encoding, a channel p is represented as an ambient $p^{\mathtt{ch}}$. Communications along that channel become communications local to $p^{\mathtt{ch}}$. I/O requests on channel p are rendered by special ambients $p^{\mathtt{io}}$. These ambients move into the channel ambient $p^{\mathtt{ch}}$, where they are opened so that the requests may interact. This yields the encoding of the π-calculus operators and types in Table XII. Type environments are translated as by the clause for restriction. Thus

$$[\![p_1 : T_1 \ldots p_n : T_n]\!] \stackrel{\text{def}}{=} p_1^{\mathtt{ch}} : [\![T_1]\!]^{\mathtt{ch}}, p_1^{\mathtt{io}} : [\![T_1]\!]^{\mathtt{io}} \ldots p_n^{\mathtt{ch}} : [\![T_n]\!]^{\mathtt{ch}}, p_n^{\mathtt{io}} : [\![T_n]\!]^{\mathtt{io}}.$$

The complete encoding needs however a further encoding step, to add channel ambients for all the free names of a process:

$$\{\![P]\!\} \stackrel{\text{def}}{=} [\![P]\!] \mid \mathtt{ch}\, p_1 \mid \ldots \mid \mathtt{ch}\, p_n$$
$$\text{where } p_1 \ldots p_n \text{ are free names of } P$$

This is behaviourally the same as encoding into MA in [Cardelli and Gordon 1999c; Cardelli et al. 1999]. At the level of types, however, the SA encoding has an advantage. The typing of the encoding into MA is studied in depth in [Cardelli et al. 1999]. The problem raised is the following. A channel ambient $p^{\mathtt{ch}}$ is immobile—it never jumps into or out of another ambient—but the type system for immobility in [Cardelli et al. 1999] is unable to type $p^{\mathtt{ch}}$ as immobile. The reason is that $p^{\mathtt{ch}}$ opens the ambients such as I/O ambients $p^{\mathtt{io}}$, which are mobile (they can jump, precisely into $p^{\mathtt{ch}}$); as pointed out in Example 4.2, the type system in [Cardelli et al. 1999] forbids immobile ambients from opening any mobile ambients. The problem is solved by extending the syntax and the type systems of MA with primitives for *objective moves* [Cardelli et al. 1999].

In SA, by contrast, coactions and ST types show that the I/O ambients $p^{\mathtt{io}}$ are opened only *after* their jump and that, when opened, they liberate processes that are immobile. The channel ambient $p^{\mathtt{ch}}$ can therefore be typed as immobile, in the

Table XII. Encoding of π-calculus processes and types

$$\mathtt{ch}\, p \stackrel{\text{def}}{=} p^{\mathtt{ch}}[\,!\overline{\mathtt{in}}\, p^{\mathtt{ch}} \mid !\overline{\mathtt{out}}\, p^{\mathtt{ch}} \mid !\mathtt{open}\, p^{\mathtt{io}}\,]$$

$$\begin{aligned}
&[\![p(q_1 : T_1 \,.\,.\, q_n : T_n).\, P]\!] \stackrel{\text{def}}{=} \\
&\qquad (\boldsymbol{\nu} k : \mathtt{BAmb}\,[\mathtt{Shh}]) \\
&\qquad (p^{\mathtt{io}}[\ \mathtt{in}\, p^{\mathtt{ch}}.\, \overline{\mathtt{open}}\, p^{\mathtt{io}}. \\
&\qquad\qquad (q_1^{\mathtt{ch}} : [\![T_1]\!]^{\mathtt{ch}}, q_1^{\mathtt{io}} : [\![T_1]\!]^{\mathtt{io}} \,.\,.\, q_n^{\mathtt{ch}} : [\![T_n]\!]^{\mathtt{ch}}, q_n^{\mathtt{io}} : [\![T_n]\!]^{\mathtt{io}})\ \ (\, k[\,\mathtt{out}\, p^{\mathtt{ch}}.\, \overline{\mathtt{open}}\, k.\, [\![P]\!]\,]\,)\ ] \\
&\qquad \mid \mathtt{open}\, k)
\end{aligned}$$

$$\begin{aligned}
[\![\overline{p}\langle q_1 \,.\,.\, q_n\rangle]\!] &\stackrel{\text{def}}{=} p^{\mathtt{io}}[\,\mathtt{in}\, p^{\mathtt{ch}}.\, \overline{\mathtt{open}}\, p^{\mathtt{io}}.\, \langle q_1^{\mathtt{ch}}, q_1^{\mathtt{io}} \,.\,.\, q_n^{\mathtt{ch}}, q_n^{\mathtt{io}}\rangle\,] \\
[\![(\boldsymbol{\nu} p : T)\, P]\!] &\stackrel{\text{def}}{=} (\boldsymbol{\nu} p^{\mathtt{ch}} : [\![T]\!]^{\mathtt{ch}})\, (\boldsymbol{\nu} p^{\mathtt{io}} : [\![T]\!]^{\mathtt{io}})\, (\mathtt{ch}\, p \mid [\![P]\!]) \\
[\![P \mid Q]\!] &\stackrel{\text{def}}{=} [\![P]\!] \mid [\![Q]\!] \\
[\![!P]\!] &\stackrel{\text{def}}{=} ![\![P]\!] \\
[\![\mathbf{0}]\!] &\stackrel{\text{def}}{=} \mathbf{0} \\
[\![Ch(T_1 \,.\,.\, T_n)]\!]^{\mathtt{ch}} &\stackrel{\text{def}}{=} \mathtt{IAmb}\,[[\![T]\!]_1^{\mathtt{ch}} \times [\![T]\!]_1^{\mathtt{io}} \times \,.\,.\, \times [\![T]\!]_n^{\mathtt{ch}} \times [\![T]\!]_n^{\mathtt{io}}] \\
[\![Ch(T_1 \,.\,.\, T_n)]\!]^{\mathtt{io}} &\stackrel{\text{def}}{=} \mathtt{STAmb}\ \dagger\ [\dagger\ [\![T]\!]_1^{\mathtt{ch}} \times [\![T]\!]_1^{\mathtt{io}} \times \,.\,.\, \times [\![T]\!]_n^{\mathtt{ch}} \times [\![T]\!]_n^{\mathtt{io}}]
\end{aligned}$$

same way as the ambient n in Example 4.2 can. Primitives for objective moves are not needed.

LEMMA 8.1. *If* $\Gamma \vdash P$, *then* $[\![\Gamma]\!] \vdash \{\![P]\!\}\ : \mathtt{BProc}\,[\mathtt{Shh}]$.

PROOF. Use the definition of $\{\![P]\!\}$ in terms of $[\![P]\!]$, and then proceed by induction on the structure of P. □

Encoding of channels, revisited. The previous π-calculus encoding has two drawbacks: (1) it is not purely compositional, for it is is defined in two steps; (2) each channel needs a centralized channel ambient, which handles all I/O requests along that channel. As pointed out in [Cardelli and Gordon 1999c], whereas such a centralization point may be acceptable for the π-calculus encoding, it is not as an encoding of communications along channels, for it forces all I/O requests on that channel to be inside the same ambient as the channel ambient. This is in general quite restrictive, because ambients form a hierarchical structure, in which I/O requests may be arbitrarily deep. Table XIII shows an encoding that does not suffer these problems; the solution adopted in [Cardelli and Gordon 1999c]—called the coalescing encoding—is behaviourally quite different. Both input and output requests along channel p are represented by ambients called p. The output ambient enters the input one, where it is opened, so that the communication takes place. After that, the continuation of the input exits and is activated.

This encoding, as the previous one, respects typing: if $\Gamma \vdash P$, then

$$[\![\Gamma]\!] \vdash [\![P]\!]\ : \mathtt{BProc}\,[\mathtt{Shh}]\,.$$

Table XIII. Encoding of π-calculus processes and types, revisited

$$
\begin{array}{rcl}
[\![p(q_1 : T_1 \mathinner{.\,.} q_n : T_n).\, P]\!] & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} k : \texttt{BAmb}\,[\texttt{Shh}]) \\
& & (p[\,\overline{\texttt{in}}\,p.\,\texttt{open}\,p.\,(q_1 : [\![T_1]\!] \mathinner{.\,.} q_n : [\![T_n]\!])\,(k[\,\texttt{out}\,p.\,\overline{\texttt{open}}\,k.\,[\![P]\!]\,] \mid \overline{\texttt{out}}\,p)\,] \\
& & \mid \texttt{open}\,k) \\
[\![\overline{p}\langle q_1 \mathinner{.\,.} q_n \rangle]\!] & \stackrel{\text{def}}{=} & p[\,\texttt{in}\,p.\,\overline{\texttt{open}}\,p.\,\langle q_1 \mathinner{.\,.} q_n \rangle\,] \\
[\![(\boldsymbol{\nu} p : T)\, P]\!] & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} p : [\![T]\!])\, [\![P]\!] \\
[\![P \mid Q]\!] & \stackrel{\text{def}}{=} & [\![P]\!] \mid [\![Q]\!] \\
[\![!P]\!] & \stackrel{\text{def}}{=} & ![\![P]\!] \\
[\![\mathbf{0}]\!] & \stackrel{\text{def}}{=} & \mathbf{0} \\
[\![Ch(T_1 \mathinner{.\,.} T_n)]\!] & \stackrel{\text{def}}{=} & \texttt{STAmb} \uparrow [\uparrow [\![T_1]\!] \times \mathinner{.\,.} \times [\![T_n]\!]]
\end{array}
$$

Since the new encoding is compositional and more direct, its operational correctness is (much) simpler to prove.

THEOREM 8.2.

(1) *If* $P \longrightarrow P'$ *then* $[\![P]\!] \Longrightarrow\simeq [\![P']\!]$;
(2) *If* $[\![P]\!] \longrightarrow Q$ *then there is* P' *such that* $P \longrightarrow P'$ *and* $Q \simeq [\![P']\!]$.

PROOF. Induction on the structure of P. The occurrences of $\simeq$ are obtained from the algebraic laws of Lemma 7.1. □

The theorem can be extended to weak reductions. From this and compositionality of the encoding we derive soundness, which says that if two target SA processes are behaviourally equivalent, then also the two source π-calculus processes are so. (These results hold both on typed and untyped calculi.) Using the same symbol for typed behavioural equivalence (barbed congruence) in the π-calculus, we have:

THEOREM 8.3. *Suppose* $\Gamma \vdash P$ *and* $\Gamma \vdash Q$. *Then* $[\![\Gamma]\!] \triangleright [\![P]\!] \simeq [\![Q]\!] : \texttt{BProc}\,[\texttt{Shh}]$ *implies* $\Gamma \triangleright P \simeq Q$.

The clause for input prefix of the encoding may be simplified as below, where the input ambient p is immediately destroyed after the communication:

$$
\begin{array}{rcl}
[\![p(q_1 : T_1 \mathinner{.\,.} q_n : T_n).\, P]\!] & \stackrel{\text{def}}{=} & p[\,\overline{\texttt{in}}\,p.\,\texttt{open}\,p.\,(q_1 : [\![T_1]\!] \mathinner{.\,.} q_n : [\![T_n]\!]).\,\overline{\texttt{open}}\,p.\,[\![P]\!]\,] \\
& & \mid \texttt{open}\,p.
\end{array}
$$

Analogous results holds using this clause for input in the encoding, but more sophisticated type systems are needed, whose formalisation we leave for future work.

In the encoding of Table XIII, coactions guarantee that exactly one output ambient enters an input one and, in the simpler input clause above, they control when the input ambient is opened.

9. RELATED WORK AND CONCLUSIONS

The introduction of coactions has been inspired by the labeled transition semantics of MA [Cardelli and Gordon 1998b], in which objects similar to coactions appear in the labels. Interaction by means of coactions also reminds us of interaction in CCS [Milner 1989] and π-calculus [Sangiorgi and Walker 2001], and distributed extensions of them, especially Seal [Vitek and Castagna 1999], but also Join, Dπ, etc. [Fournet et al. 1996; Hennessy and Riely 1998]. Our type system for immobility in SA is similar to the one for MA [Cardelli et al. 1999]. In the latter, however, immobility does not implies persistence; a separate annotation records whether an ambient may be opened or not. If needed, such a distinction can also be added to our system. By contrast, the ST (single-threadness) type system is specific to SA: coactions are necessary to make sense of it. The ST system could be formulated on MA, but all properties it gives us, such as removal of grave interferences, algebraic laws, more precise information on mobility, would be lost. (It might be reasonable, by contrast, to adapt the ST type system to Seal.) Single-threadness is a form of linearity, hence the typing rules borrow ideas from ordinary type systems with linear types (on λ-calculi or elsewhere).

The algebraic laws of Section 7 are inspired by the algebraic theory of process calculi such as the π-calculus. Both in typed SA and in the π-calculus, the laws only require conditions to control plain interferences. As mentioned in the Introduction, and in the discussions of the examples of Section 8, the algebraic theory of MA appears to be rather poor, due to the interference problems. Even the MA analogues of the very simple, untyped, laws of Lemma 7.1 would be hard to formulate. For instance, consider law (5) of the lemma. Removing coactions it becomes $n[\,P \mid m[\,\mathtt{out}\,n.\,Q\,]\,] \simeq m[\,Q\,] \mid n[\,P\,]$. This equality may be broken in MA if: (i) P uses a capability; (ii) an ambient in P tries to enter m; (iii) an ambient in the external environment tries to enter n and then m; (iv) a process in the external environment tries to open n. Furthermore, maintaining these invariances through reduction needs further conditions; for instance if $P = k[\,\mathtt{open}\,h.\,\mathtt{out}\,n.\,\mathtt{open}\,n\,]$, then an ambient h from the external environment could enter n and k, be opened; k could then exit n, be opened, and open n. Types to control mobility in MA [Cardelli et al. 1999] may be of limited help: they may not be precise enough, as in the example of encoding of channels; they do not rule out various grave interferences (examples: in the law above, declaring n immobile and not-openable does not prevent (ii) and (iii); immobility is of no help for the interference problems discussed in the renaming and routing protocols of Section 8).

Because of coactions, the movement interactions of SA are synchronous. Hence synchronisation mechanisms between potential interacting partners will be required in distributed implementations. Types, including those in this paper, would certainly help. Other tuning might be appropriate, perhaps along the lines of the Join primitives [Fournet et al. 1996]. Synchronisations are also (implicitly) necessary in MA: for instance in all cases of grave interference, and on interfering opening actions.

We discuss some possible refinements of the ST and immobility type systems. The rules for ST could be relaxed so to allow more than one capability to be exercised, in parallel, at the top level. For instance, an $\mathtt{open}\,n$ capability does not really needs the thread if ambient n does not require the thread when opened. Thus we could type as ST an ambient such as $m[\,\mathtt{in}\,n.\,P_1 \mid \mathtt{open}\,n \mid n[\,P\,]\,]$. For this, we would need a new type to describe capabilities, such as $\mathtt{open}\,n$, that can be used without the thread (and of course leaves no thread).

At present we are very drastic about ST ambients opened inside non-ST ambients: the liberated process should bear no thread. We could be more flexible, allowing the liberated process to be of the same "class" as the enclosing ambient. For instance, an ST ambient opened inside an immobile ambient would liberate an immobile process.

The type systems allow rather limited forms of subtyping (Lemmas 5.4 and 5.5): an ST process without thread may be also typed as ST with thread, as basic, as immobile. Moreover, to obtain this we added the constrain on recursive processes that the recursion variable does not appear free underneath an ambient. Subtyping could be developed further: removing the constrain on recursion; but also allowing other forms of subtyping (for instance, allowing any ST or immobile ambient to be typed also as a basic ambient).

In this paper we have ignored deadlocks. For instance, we can type processes like $n[\,\mathbf{0}\,]$, or $n[\,m[\,P\,]\,]$, or $n[\,\overline{\mathtt{out}}\,m.\,P\,]$, or $n[\,\overline{\mathtt{open}}\,m.\,P\,]$ (assuming P itself types), although they are all deadlocked. We consider deadlock an orthogonal issue. Our objective here was to find *proof techniques* (in particular, algebraic laws) for behavioural equivalence of programs. Therefore our objective is not preventing deadlock but, for instance, proving that a deadlocked process is equivalent to $\mathbf{0}$. We leave the study of techniques for preventing deadlock for future work.

We have mentioned that MA can be encoded into SA. We doubt that the converse is possible, especially for typed SA, but we leave proving or disproving it for future work. Another challenging expressiveness issue is the comparison with Seal [Vitek and Castagna 1999].

By the time the full version of this paper has been completed, a few papers have appeared that treat SA. Sangiorgi and Valente [2001] have carried out a distributed implementation of SA. Dezani-Ciancaglini and Salvo [2000], Bugliesi and Castagna [2001], and Degano, Levi, and Bodei [2000] address security issues for SA by using static techniques, in particular types and control-flow analysis. The proposed methods check that the use of capabilities and the movement of ambients respect some security policies, based mostly on a given classification of ambient names. Coactions naturally give protection against the attacks attempting to break those security policies. Assume for instance we want to guarantee that an ambient, say n, is opened (thus giving access to its internal processes) only inside ambients of a trusted domain. If coaction $\overline{\mathtt{open}}\,n$ is properly used, ambient n may cross untrusted ambients (possibly knowing its name) without being opened. Zimmer [2000] has studied the expressiveness of pure SA, that is, SA without communications, showing how communication as in the π-calculus can be simulated by means of modifications of the hierarchical structure of an ambient system. Guan, Yang, and You [2000] have proposed a variant of SA with a different syntax for coactions: in $\overline{\mathtt{in}}\,n$ and $\overline{\mathtt{out}}\,n$, the name n refers to the ambient that moves, rather than the ambient than allows the move, and $\overline{\mathtt{open}}$ has no name parameter. For this variant of SA without communication the same authors [Guan et al. 2001] modify our type system for basic, ST and immobility types to allow a more flexible typing of $\mathtt{open}$ and $\overline{\mathtt{open}}$. For instance an ambient can open another ambient with not exactly the same type (see the discussion above about an immobile ambients which open an ST ambient that liberates an immobile process). Amtoft et al. [2001] consider an higher-order version of MA with the capability $\overline{\mathtt{open}}$. They suggest different kinds of polymorphism in such a calculus and design a type system accordingly. The case of so called "orderly-communication" allows to type ambients in which the type of the exchanged data can change over time. It would be interesting to add this feature also to our types. For instance the variant of the second encoding of π-calculus in Section 8 could probably be typable using "orderly-communication". Finally, Merro and Hennessy [2002] have developed a labeled bisimilarity for (a variant of) Safe Ambients, and proved that it coincides with barbed congruence.

Acknowledgments. We have benefited from discussions with Luca Cardelli, Giorgio Ghelli, and Andy Gordon. Sergio Maffeis pointed out the need of recursive types in one of the examples. Xudong Guan suggested an improvement in the example of renaming.

We are very grateful to three anonymous referees, for their valuable comments and suggestions: the resulting modifications have improved a lot the paper.

APPENDIX

A. INTERFERENCES

We give a definition of grave and plain interferences in SA as a list of all possible syntactic patterns producing an interference. Simply removing coactions, we obtain the corresponding list of plain and grave interferences for MA. If L is a set of simple capabilities $\mu_1, \ldots, \mu_n$, then we write $P^{\{L\}}$ if $P = \mu_1 .\, P_1 \mid \ldots \mid \mu_n .\, P_n \mid Q$, for some $P_1, \ldots, P_n, Q$.

We say that a context C is *active* iff the hole is unguarded and not in the body of recursion.

Definition A.1. **(plain interferences)** Let $P \in SA$. Process P *has a plain interference pattern* iff $P \equiv C[I]$, for some active context C, and the form of I is one of the following:

(1) $n[\, P^{\{\mathtt{in}\, m\}} \,] \mid m[\, Q^{\{\overline{\mathtt{in}}\, m\}} \,] \mid m[\, R^{\{\overline{\mathtt{in}}\, m\}} \,]$;
(2) $n[\, P^{\{\mathtt{in}\, m, \mathtt{in}\, m\}} \,] \mid m[\, Q^{\{\overline{\mathtt{in}}\, m\}} \,]$;
(3) $n[\, P^{\{\mathtt{in}\, m\}} \,] \mid h[\, R^{\{\mathtt{in}\, m\}} \,] \mid m[\, Q^{\{\overline{\mathtt{in}}\, m\}} \,]$;
(4) $n[\, P^{\{\mathtt{in}\, m\}} \,] \mid m[\, Q^{\{\overline{\mathtt{in}}\, m, \overline{\mathtt{in}}\, m\}} \,]$;
(5) $m[\, n[\, P^{\{\mathsf{out}\, m, \mathsf{out}\, m\}} \,] \mid R^{\{\overline{\mathtt{out}}\, m\}} \,]$;
(6) $m[\, n[\, P^{\{\mathsf{out}\, m\}} \,] \mid R^{\{\overline{\mathtt{out}}\, m, \overline{\mathtt{out}}\, m\}} \,]$;
(7) $m[\, n[\, P^{\{\mathsf{out}\, m\}} \,] \mid h[\, Q^{\{\mathsf{out}\, m\}} \,] \mid R^{\{\overline{\mathtt{out}}\, m\}} \,]$;
(8) $\mathsf{open}\, n .\, P' \mid \mathsf{open}\, n .\, P \mid n[\, Q^{\{\overline{\mathtt{open}}\, n\}} \,]$;
(9) $\mathsf{open}\, n .\, P \mid n[\, Q^{\{\overline{\mathtt{open}}\, n\}} \,] \mid n[\, R^{\{\overline{\mathtt{open}}\, n\}} \,]$;
(10) $\mathsf{open}\, n .\, P \mid n[\, Q^{\{\overline{\mathtt{open}}\, n, \overline{\mathtt{open}}\, n\}} \,]$.

Definition A.2. **(grave interferences)** Let $P \in SA$. Process P *has a grave interference pattern* iff $P \equiv C[I]$, for some active context C, and the form of I is one of the following:

(1) $\mathsf{open}\, n .\, R \mid n[\, P^{\{\mathtt{in}\, m, \overline{\mathtt{open}}\, n\}} \,] \mid m[\, Q^{\{\overline{\mathtt{in}}\, m\}} \,]$;
(2) $\mathsf{open}\, m .\, R \mid n[\, P^{\{\mathtt{in}\, m\}} \,] \mid m[\, Q^{\{\overline{\mathtt{in}}\, m, \overline{\mathtt{open}}\, m\}} \,]$;
(3) $n[\, P^{\{\mathtt{in}\, m, \mathtt{in}\, h\}} \,] \mid m[\, Q^{\{\overline{\mathtt{in}}\, m\}} \,] \mid h[\, R^{\{\overline{\mathtt{in}}\, h\}} \,]$;
(4) $n[\, P^{\{\mathtt{in}\, m, \mathtt{in}\, m\}} \,] \mid m[\, Q^{\{\overline{\mathtt{in}}\, m, \overline{\mathtt{in}}\, m\}} \,]$;
(5) $n[\, P^{\{\mathtt{in}\, m\}} \,] \mid m[\, Q^{\{\overline{\mathtt{in}}\, m, \mathtt{in}\, h\}} \,] \mid h[\, R^{\{\overline{\mathtt{in}}\, h\}} \,]$;

(6) $h[\,R^{\{\overline{\mathsf{out}}\,h\}} \mid n[\,P^{\{\mathsf{in}\,m,\mathsf{out}\,h\}}\,] \mid m[\,Q^{\{\overline{\mathsf{in}}\,m\}}\,]\,]$;
(7) $h[\,R^{\{\overline{\mathsf{out}}\,h\}} \mid n[\,P^{\{\mathsf{in}\,m\}}\,] \mid m[\,Q^{\{\overline{\mathsf{in}}\,m,\mathsf{out}\,h\}}\,]\,]$;
(8) $n[\,P^{\{\mathsf{in}\,m,\overline{\mathsf{out}}\,n\}} \mid h[\,R^{\{\mathsf{out}\,n\}}\,]\,] \mid m[\,Q^{\{\overline{\mathsf{in}}\,m\}}\,]$;
(9) $\mathsf{open}\,m.\,Q \mid m[\,R^{\{\overline{\mathsf{open}}\,m,\overline{\mathsf{out}}\,m\}} \mid n[\,P^{\{\mathsf{out}\,m\}}\,]\,]$;
(10) $m[\,R^{\{\overline{\mathsf{out}}\,m,\mathsf{open}\,n\}} \mid n[\,P^{\{\mathsf{out}\,m,\overline{\mathsf{open}}\,n\}}\,]\,]$;
(11) $h[\,R^{\{\overline{\mathsf{out}}\,h\}} \mid m[\,Q^{\{\overline{\mathsf{out}}\,m,\mathsf{out}\,h\}} \mid n[\,P^{\{\mathsf{out}\,m\}}\,]\,]\,]$;
(12) $\mathsf{open}\,n.\,P \mid \mathsf{open}\,n.\,R \mid n[\,Q^{\{\overline{\mathsf{open}}\,n,\overline{\mathsf{open}}\,n\}}\,]$;
(13) $m[\,n[\,P^{\{\mathsf{out}\,m,\mathsf{out}\,m\}}\,] \mid Q^{\{\overline{\mathsf{out}}\,m,\overline{\mathsf{out}}\,m\}}\,]$.

It is worth explaining some particular cases of Definitions A.1 and A.2. The process

$$\mathsf{open}\,n.\,P_1 \mid \mathsf{open}\,n.\,P_2 \mid n[\,\overline{\mathsf{open}}\,n.\,P_3 \mid \overline{\mathsf{open}}\,n.\,P_4\,]$$

has several plain interference patterns corresponding to cases (8) and (10) of Definition A.1. It has however also a grave interference pattern corresponding to case (12) of Definition A.2. This is consistent with Definition 3.3 of Section 3: the corresponding well-formed colored process has indeed both kinds of interferences. For instance the two overlapping reductions corresponding to a plain interference (case (10) of Definition A.1) are

$$\mathsf{open}\,n_{\chi_1}.\,P_1 \mid \mathsf{open}\,n_{\chi_2}.\,P_2 \mid n[\,\overline{\mathsf{open}}\,n_{\chi_3}.\,P_3 \mid \overline{\mathsf{open}}\,n_{\chi_4}.\,P_4\,] \longrightarrow_{\{\chi_2,\chi_3\}}$$
$$\mathsf{open}\,n_{\chi_1}.\,P_1 \mid P_2 \mid Q_3 \mid \overline{\mathsf{open}}\,n_{\chi_4}.\,P_4$$
$$\mathsf{open}\,n_{\chi_1}.\,P_1 \mid \mathsf{open}\,n_{\chi_2}.\,P_2 \mid n[\,\overline{\mathsf{open}}\,n_{\chi_3}.\,P_3 \mid \overline{\mathsf{open}}\,n_{\chi_4}.\,P_4\,] \longrightarrow_{\{\chi_2,\chi_4\}}$$
$$\mathsf{open}\,n_{\chi_1}.\,P_1 \mid P_2 \mid \overline{\mathsf{open}}\,n_{\chi_3}.\,P_3 \mid P_4.$$

The two non-overlapping reductions corresponding to a grave interference (case (12) of Definition A.2) are

$$\mathsf{open}\,n_{\chi_1}.\,P_1 \mid \mathsf{open}\,n_{\chi_2}.\,P_2 \mid n[\,\overline{\mathsf{open}}\,n_{\chi_3}.\,P_3 \mid \overline{\mathsf{open}}\,n_{\chi_4}.\,P_4\,] \longrightarrow_{\{\chi_1,\chi_3\}} \tag{14}$$
$$P_1 \mid \mathsf{open}\,n_{\chi_2}.\,P_2 \mid P_3 \mid \overline{\mathsf{open}}\,n_{\chi_4}.\,P_4$$
$$\mathsf{open}\,n_{\chi_1}.\,P_1 \mid \mathsf{open}\,n_{\chi_2}.\,P_2 \mid n[\,\overline{\mathsf{open}}\,n_{\chi_3}.\,P_3 \mid \overline{\mathsf{open}}\,n_{\chi_4}.\,P_4\,] \longrightarrow_{\{\chi_2,\chi_4\}} \tag{15}$$
$$\mathsf{open}\,n_{\chi_1}.\,P_1 \mid P_2 \mid \overline{\mathsf{open}}\,n_{\chi_3}.\,P_3 \mid P_4.$$

One may argue that case (12) of Definition A.2 should be considered as a plain interference, for there is no conflict on the final global addresses of the processes involved. Similarly for cases (3) (assuming $h = m$), (4), and (13) of Definition A.2. Definition 3.3 of Section 3 could then be adapted accordingly. We prefer, however, to stick to 'overlapping redexes' as the criterion for distinguishing between plain and grave interferences. One reason is simplicity. Another reason is that interferences such as (14-15) above, as all interferences resulting from non-overlapping redexes, give problems anyway. For instance in distributed implementations (see the discussion in the Introduction), and in theories based on non-interleaving semantics (some preliminary work in this direction is [Gadducci and Montanari 2001]). Hence we want to make sure that all such interferences are removed through ST and immobility types.

Reductions (14-15) may look similar to the following reductions of the CCS process $\overline{x}.\,P_1 \mid \overline{x}.\,P_2 \mid x.\,P_3 \mid x.\,P_4$:

$$\overline{x}.\,P_1 \mid \overline{x}.\,P_2 \mid x.\,P_3 \mid x.\,P_4 \longrightarrow P_1 \mid \overline{x}.\,P_2 \mid P_3 \mid x.\,P_4$$

$$\overline{x}.\,P_1 \mid \overline{x}.\,P_2 \mid x.\,P_3 \mid x.\,P_4 \longrightarrow \overline{x}.\,P_1 \mid P_2 \mid x.\,P_3 \mid P_4$$

This is however misleading, for the above CCS reductions do not interfere – one redex that fires does not destroy the other.

The remainder of the section uses the terminology and the notations introduced in Section 3. Lemma A.6 shows that the definition above of plain and grave interferences is equivalent to Definition 3.3 of Section 3 (which uses Definition 3.2 over CSA).

Lemma A.3. *Let $P \in$ CSA. If P has an interference on $\mathcal{C}_1$ and $\mathcal{C}_2$ and $P \equiv P'$, then also P' has an interference on $\mathcal{C}_1$ and $\mathcal{C}_2$.*

Lemma A.4. *Let $P, Q \in SA$ such that $P \equiv Q$. For any well-formed colored process $Q^\star$ such that $Q^\star_{\text{erase}} = Q$ there exists a well-formed colored process $P^\star$ such that $P^\star_{\text{erase}} = P$ and $P^\star \equiv Q^\star$.*

Lemma A.5. *Let $P, Q \in$ CSA such that $P \equiv Q$. We have $P_{\text{erase}} \equiv Q_{\text{erase}}$.*

Lemma A.6. *An SA process P has a plain interference (Definition 3.3) iff it has a plain interference pattern (Definition A.1). Similarly, P has a grave interference iff it has a grave interference pattern (Definition A.2).*

Proof. One implication of the proof is simple. Assume P is an SA process which has a plain interference pattern (Definition A.1). It means that $P \equiv C[I]$, where C is an active context and I takes one of the forms listed in Definition A.1. Let $I^\star$ be a well-formed colored process such that $I^\star_{\text{erase}} = I$. It is easy to check that, in any case of Definition A.1, we have two transitions $I^\star \longrightarrow_{\mathcal{C}_1} I_1$ and $I^\star \longrightarrow_{\mathcal{C}_2} I_2$, which interfere on $\mathcal{C}_1$ and $\mathcal{C}_2$ with $\mathcal{C}_1 \cap \mathcal{C}_2 \neq \emptyset$.

We then observe that, since C is an active context, there exists a well-formed colored context $C^\star$ such that $C^\star[I^\star]$ is a well-formed colored process. Moreover, arguing by induction on the context $C^\star$, we have two transitions which interfere on $\mathcal{C}_1$ and $\mathcal{C}_2$

$$C^\star[I^\star] \longrightarrow_{\mathcal{C}_1} C^\star[I_1]$$

$$C^\star[I^\star] \longrightarrow_{\mathcal{C}_2} C^\star[I_2].$$

By Lemma A.4 there exists a well-formed colored process $P^\star$ such that $P^\star \equiv C^\star[I^\star]$ and $P^\star_{\text{erase}} = P$. Therefore, by Lemma A.3, also $P^\star$ has a plain interference, and thus P has a plain interference. Similarly for the case when P has a grave interference pattern (Definition A.2).

The converse case is rather complex. Assume P is an SA process which has an interference (Definition 3.3). It means that a well-formed colored process $P^\star$, such that $P^\star_{\mathrm{erase}} = P$, has an interference. By Definition 3.2 there are two transitions $P^\star \longrightarrow_{\mathcal{C}_1} P_1$ and $P^\star \longrightarrow_{\mathcal{C}_2} P_2$ and there is no P_3, such that $P_1 \longrightarrow_{\mathcal{C}_2} P_3$ and $P_2 \longrightarrow_{\mathcal{C}_1} P_3$.

We notice that $\mathcal{C}_1$ and $\mathcal{C}_2$ are non-empty set of colors, meaning that some colored capabilities have been consumed. Therefore, arguing by induction on the depth of the inference of $P^\star \longrightarrow_{\mathcal{C}_1} P_1$, we derive that $P^\star \equiv C_1[Q_1]$ and $P_1 \equiv C_1[Q_1']$ for some well-formed colored context C_1 and well-formed colored processes Q_1 and Q_1'. Moreover, we have a transition

$$Q_1 \longrightarrow_{\mathcal{C}_1} Q_1'$$

which has been obtained by applying either rule CR-In, or rule CR-Out, or rule CR-Open.

We now apply a similar argument to transition $P^\star \longrightarrow_{\mathcal{C}_2} P_2$ and we derive that $P^\star \equiv C_2[Q_2]$ and $P_2 \equiv C_2[Q_2']$ for some well-formed colored processes Q_2 and Q_2' and well-formed colored context C_2. Moreover, we have a transition

$$Q_2 \longrightarrow_{\mathcal{C}_2} Q_2'$$

which has been obtained by applying either rule CR-In, or rule CR-Out, or rule CR-Open. We observe also that the holes of C_1 and C_2 are not under recursion and are unguarded, meaning that both $C_{1\mathrm{erase}}$ and $C_{2\mathrm{erase}}$ are active contexts.

The proof proceeds by considering all possible combinations of rules CR-In, CR-Out and CR-Open applied to derive transitions $Q_1 \longrightarrow_{\mathcal{C}_1} Q_1'$ and $Q_2 \longrightarrow_{\mathcal{C}_2} Q_2'$. To simplify the discussion we consider separately the case of plain and grave interferences.

Plain Interferences We have $\mathcal{C}_1 \cap \mathcal{C}_2 \neq \emptyset$. As a consequence, both transition $Q_1 \longrightarrow_{\mathcal{C}_1} Q_1'$ and $Q_2 \longrightarrow_{\mathcal{C}_2} Q_2'$ have been obtained by the same kind of rule, either by rule CR-In, or by rule CR-Out, or by rule CR-Open. In the following let $\mathcal{C}_1 = \{\chi, \chi_1\}$ and $\mathcal{C}_2 = \{\chi, \chi_2\}$.

(1) Assume that rule CR-In has been applied. There are two cases, depending on whether χ is the color of a capability $\mathtt{in}\, m$ or of a capability $\overline{\mathtt{in}}\, m$ for some m.

Assume that χ is the color of a capability $\mathtt{in}\, m$. It means that χ_1 and χ_2 are the colors of two capabilities $\overline{\mathtt{in}}\, m$. Therefore, we have

$$Q_1 \longrightarrow_{\{\chi,\chi_1\}} m[\, n[\, R_1 \mid R_2\,] \mid S_1 \mid S_2\,]$$

$$Q_2 \longrightarrow_{\{\chi,\chi_2\}} m[\, n[\, R_1 \mid R_2\,] \mid T_1 \mid T_2\,]$$

where

$$Q_1 = n[\, \mathtt{in}\, m_\chi . R_1 \mid R_2\,] \mid m[\, \overline{\mathtt{in}}\, m_{\chi_1} . S_1 \mid S_2\,]$$

$$Q_2 = n[\, \mathtt{in}\, m_\chi . R_1 \mid R_2\,] \mid m[\, \overline{\mathtt{in}}\, m_{\chi_2} . T_1 \mid T_2\,].$$

There are now two possibilities: either $\overline{\mathtt{in}}\, m_{\chi_1}$ and $\overline{\mathtt{in}}\, m_{\chi_2}$ are contained in the same ambient m, or they are contained in two different ambients m. It means that $P^\star \equiv C[I]$, where C_{erase} is an active context and I is one of the following processes

—$n[\, \mathtt{in}\, m_\chi . R_1 \mid R_2\,] \mid m[\, \overline{\mathtt{in}}\, m_{\chi_1} . S_1 \mid \overline{\mathtt{in}}\, m_{\chi_2} . T_1 \mid S\,]$;

—$n[\,\mathtt{in}\, m_\chi.\, R_1 \mid R_2\,] \mid m[\,\overline{\mathtt{in}}\, m_{\chi_1}.\, S_1 \mid S_2\,] \mid m[\,\overline{\mathtt{in}}\, m_{\chi_2}.\, T_1 \mid T_2\,]$.
In both cases $C[I]_{\mathrm{erase}}$ has a plain interference pattern: cases (4) and (1) of Definition A.1. By Lemma A.5 we have $P^\star_{\mathrm{erase}} \equiv C[I]_{\mathrm{erase}}$ showing that also P has a plain interference pattern.
The case when χ is the color of a capability $\overline{\mathtt{in}}\, m$ is similar.

(2) Assume that rule CR-Open has been applied. There are two cases, depending on whether χ is the color of a capability $\mathtt{open}\, n$ or of a capability $\overline{\mathtt{open}}\, n$ for some ambient n.
Assume that χ is the color of a capability $\mathtt{open}\, n$. It means that χ_1 and χ_2 are the colors of two capabilities $\overline{\mathtt{open}}\, n$. Therefore, we have

$$Q_1 \longrightarrow_{\{\chi,\chi_1\}} S \mid R_1 \mid R_2$$

$$Q_2 \longrightarrow_{\{\chi,\chi_2\}} S \mid S_1 \mid S_2$$

where

$$Q_1 = \mathtt{open}\, n_\chi.\, S \mid n[\,\overline{\mathtt{open}}\, n_{\chi_1}.\, R_1 \mid R_2\,]$$

$$Q_2 = \mathtt{open}\, n_\chi.\, S \mid n[\,\overline{\mathtt{open}}\, n_{\chi_2}.\, S_1 \mid S_2\,].$$

There are two possibilities: either $\overline{\mathtt{open}}\, n_{\chi_1}$ and $\overline{\mathtt{open}}\, n_{\chi_2}$ are contained in the same ambient n, or they are contained in two different ambients n. It means that $P^\star \equiv C[I]$ where C_{erase} is an active context and I is one of the following processes
—$\mathtt{open}\, n_\chi.\, S \mid n[\,\overline{\mathtt{open}}\, n_{\chi_1}.\, R_1 \mid \overline{\mathtt{open}}\, n_{\chi_2}.\, S_1 \mid S\,]$;
—$\mathtt{open}\, n_\chi.\, S \mid n[\,\overline{\mathtt{open}}\, n_{\chi_1}.\, R_1 \mid R_2\,] \mid n[\,\overline{\mathtt{open}}\, n_{\chi_2}.\, S_1 \mid S_2\,]$.
In both cases $C[I]_{\mathrm{erase}}$ has a plain interference pattern: cases (10) and (9) of Definition A.1. By Lemma A.5 we have $P^\star_{\mathrm{erase}} \equiv C[I]_{\mathrm{erase}}$ showing that also P has a plain interference pattern.
The case when χ is the color of a capability $\overline{\mathtt{open}}\, n$ is similar.

(3) The case when rule CR-Out has been applied is similar to that of CR-In.

Grave Interferences We have $\mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset$. In the following let $\mathcal{C}_1 = \{\chi_1, \chi_2\}$ and $\mathcal{C}_2 = \{\chi_3, \chi_4\}$.

(1) (CR-Open,CR-In). Assume that $Q_1 \longrightarrow_{\mathcal{C}_1} Q_1'$ has been obtained by applying rule CR-Open and that $Q_2 \longrightarrow_{\mathcal{C}_2} Q_2'$ has been obtained by applying rule CR-In. Therefore, we have

$$Q_1 \longrightarrow_{\{\chi_1,\chi_2\}} R_1 \mid R_2 \mid R_3$$

$$Q_2 \longrightarrow_{\{\chi_3,\chi_4\}} m[\,h[\,S_1 \mid S_2\,] \mid T_1 \mid T_2\,].$$

where

$$Q_1 = \mathtt{open}\, n_{\chi_1}.\, R_1 \mid n[\,\overline{\mathtt{open}}\, n_{\chi_2}.\, R_2 \mid R_3\,]$$

$$Q_2 = h[\,\mathtt{in}\, m_{\chi_3}.\, S_1 \mid S_2\,] \mid m[\,\overline{\mathtt{in}}\, m_{\chi_4}.\, T_1 \mid T_2\,].$$

We now look at all possible shapes of C_1, C_2, Q_1 and Q_2. Using

$$P^\star \equiv C_1[Q_1] \equiv C_2[Q_2]$$

we derive that

$$C_1[Q_1] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[Q_2']$$

$$C_2[Q_2] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[Q_1'].$$

We observe that in $C_1[Q_1] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[Q_2']$ there are three cases: (i) either only Q_1 contributes to the action, or (ii) only C_1 contributes to the action or (iii) C_1 and Q_1 interact. Similarly, in $C_2[Q_2] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[Q_1']$ there are three cases: (i) either only Q_2 contributes to the action, or (ii) only C_2 contributes to the action or (iii) C_2 and Q_2 interact.
The main point of the proof consists in showing that cases (i) and (ii) cannot occur and that in case (iii) a grave interference pattern occurs. We exploit the fact that there is no process P_3 such that

$$C_1[R_1 \mid R_2 \mid R_3] \longrightarrow_{\{\chi_3,\chi_4\}} P_3$$

$$C_2[m[\,h[\,S_1 \mid S_2\,] \mid T_1 \mid T_2\,]] \longrightarrow_{\{\chi_1,\chi_2\}} P_3.$$

(i) Assume that only Q_1 contributes to the action in

$$C_1[Q_1] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[Q_2'].$$

We notice that R_1 and R_2 cannot contain Q_2 because the capabilities with colors χ_3 and χ_4 are ready to interact in $P^\star$. It means that $R_3 \equiv C_3[Q_2]$ for some context C_3 and that

$$C_3[Q_2] \longrightarrow_{\{\chi_3,\chi_4\}} C_3[Q_2'].$$

Therefore, we have

$$C_2[Q_2] \equiv C_1[\mathtt{open}\, n_{\chi_1}.\, R_1 \mid n[\,\overline{\mathtt{open}}\, n_{\chi_2}.\, R_2 \mid C_3[Q_2]\,]\,]$$

$$C_2[Q_2'] \equiv C_1[\mathtt{open}\, n_{\chi_1}.\, R_1 \mid n[\,\overline{\mathtt{open}}\, n_{\chi_2}.\, R_2 \mid C_3[Q_2']\,]\,].$$

Arguing by induction on the context, we obtain

$$C_1[Q_1] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[R_1 \mid R_2 \mid C_3[Q_2]] \longrightarrow_{\{\chi_3,\chi_4\}} C_1[R_1 \mid R_2 \mid C_3[Q_2']].$$

Similarly, we obtain also

$$C_2[Q_2] \longrightarrow_{\{\chi_3,\chi_4\}} C_1[\mathtt{open}\, n_{\chi_1}.\, R_1 \mid n[\,\overline{\mathtt{open}}\, n_{\chi_2}.\, R_2 \mid C_3[Q_2'\,]]$$

$$C_1[\mathtt{open}\, n_{\chi_1}.\, R_1 \mid n[\,\overline{\mathtt{open}}\, n_{\chi_2}.\, R_2 \mid C_3[Q_2'\,]] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[R_1 \mid R_2 \mid C_3[Q_2'].$$

This contradicts, by Lemma A.3, the fact that process $P^\star$ has an interference on $\mathcal{C}_1$ and $\mathcal{C}_2$.
Assume that only Q_2 contributes to the action in

$$C_2[Q_2] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[Q_1'].$$

We notice that S_1 and T_1 cannot contain Q_1 because the capabilities with colors χ_1 and χ_2 are ready to interact in $P^\star$. Therefore, either S_2 or T_2 contain Q_1. We show the first case only, the other is analogous. It means that $S_2 \equiv C_3[Q_1]$ for some context C_3 and that

$$C_3[Q_1] \longrightarrow_{\{\chi_1,\chi_2\}} C_3[Q_1'].$$

Therefore, we have

$$C_1[Q_1] \equiv C_2[h[\,\mathtt{in}\,m_{\chi_3}.\,S_1 \mid C_3[Q_1]\,] \mid m[\,\overline{\mathtt{in}}\,m_{\chi_4}.\,T_1 \mid T_2\,]]$$

$$C_1[Q_1'] \equiv C_2[h[\,\mathtt{in}\,m_{\chi_3}.\,S_1 \mid C_3[Q_1']\,] \mid m[\,\overline{\mathtt{in}}\,m_{\chi_4}.\,T_1 \mid T_2\,]].$$

Arguing by induction on the context, we obtain

$$C_2[Q_2] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[m[\,h[\,S_1 \mid C_3[Q_1]\,] \mid T_1 \mid T_2\,]]$$

$$C_2[m[\,h[\,S_1 \mid C_3[Q_1]\,] \mid T_1 \mid T_2\,]] \longrightarrow_{\{\chi_1,\chi_2\}} C_2[m[\,h[\,S_1 \mid C_3[Q_1']\,] \mid T_1 \mid T_2\,]].$$

Similarly, we obtain also

$$C_1[Q_1] \longrightarrow_{\{\chi_1,\chi_2\}} C_2[h[\,\mathtt{in}\,m_{\chi_3}.\,S_1 \mid C_3[Q_1']\,] \mid m[\,\overline{\mathtt{in}}\,m_{\chi_4}.\,T_1 \mid T_2\,]$$

$$C_2[h[\,\mathtt{in}\,m_{\chi_3}.\,S_1 \mid C_3[Q_1']\,] \mid m[\,\overline{\mathtt{in}}\,m_{\chi_4}.\,T_1 \mid T_2\,] \longrightarrow_{\{\chi_3,\chi_4\}}$$

$$C_2[m[\,h[\,S_1 \mid C_3[Q_1']\,] \mid T_1 \mid T_2\,]].$$

This contradicts, by Lemma A.3, the fact that process $P^\star$ has an interference on $\mathcal{C}_1$ and $\mathcal{C}_2$.

(ii) Assume that only C_1 contributes to the action in

$$C_1[Q_1] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[Q_2'].$$

It means that $C_1 \equiv C_3[Q_2]$ for some context C_3. Therefore, we have also $C_1[Q_1] \equiv C_3[Q_2][Q_1] \equiv C_2[Q_2]$, $C_1[Q_1'] \equiv C_3[Q_2][Q_1']$ and $C_2[Q_2'] \equiv C_3[Q_2'][Q_1]$. Arguing by induction on the context, we obtain

$$C_1[Q_1] \longrightarrow_{\{\chi_1,\chi_2\}} C_3[Q_2][Q_1'] \longrightarrow_{\{\chi_3,\chi_4\}} C_3[Q_2'][Q_1']$$

$$C_2[Q_2] \longrightarrow_{\{\chi_3,\chi_4\}} C_3[Q_2'][Q_1] \longrightarrow_{\{\chi_1,\chi_2\}} C_3[Q_2'][Q_1'].$$

This contradicts, by Lemma A.3, the fact that process $P^\star$ has an interference on $\mathcal{C}_1$ and $\mathcal{C}_2$.

The case when only C_2 contributes to the action in $C_2[Q_2] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[Q_1']$ is similar.

(iii) Assume that C_1 and Q_1 interact in transition

$$C_1[Q_1] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[Q_2']$$

and analogously that C_2 and Q_2 interact in transition

$$C_2[Q_2] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[Q_1'].$$

The only possibility is that ambient n is shared by the two colored reductions: either ambients h and n coincide and contain both $\overline{\mathtt{open}}\,n_{\chi_2}$ and $\mathtt{in}\,m_{\chi_3}$, or ambients m and n coincide and contain both $\overline{\mathtt{open}}\,n_{\chi_2}$ and $\overline{\mathtt{in}}\,m_{\chi_4}$.

It means that $P^* \equiv C[I]$ where C_{erase} is an active context and I is one of the following processes

—$\mathtt{open}\,n_{\chi_1}.\,R_1 \mid n[\,\mathtt{in}\,m_{\chi_3}.\,S_1 \mid \overline{\mathtt{open}}\,n_{\chi_2}.\,R_2 \mid R\,] \mid m[\,\overline{\mathtt{in}}\,m_{\chi_4}.\,T_1 \mid T_2\,]$;

—$\mathtt{open}\,n_{\chi_1}.\,R_1 \mid h[\,\mathtt{in}\,n_{\chi_3}.\,S_1 \mid S_2\,] \mid n[\,\overline{\mathtt{in}}\,n_{\chi_4}.\,T_1 \mid \overline{\mathtt{open}}\,n_{\chi_2}.\,R_2 \mid T\,]$.

In both cases $C[I]_{\mathrm{erase}}$ has a grave interference pattern: cases (1) and (2) of Definition A.2. By Lemma A.5 we have $P^{\star}_{\mathrm{erase}} \equiv C[I]_{\mathrm{erase}}$ showing that also P has a plain interference pattern.

(2) (CR-Open, CR-Open). Assume that rule CR-Open has been applied to derive both transitions. Therefore, we have

$$Q_1 \longrightarrow_{\{\chi_1,\chi_2\}} R_1 \mid R_2 \mid R_3$$

$$Q_2 \longrightarrow_{\{\chi_3,\chi_4\}} S_1 \mid S_2 \mid S_3.$$

where

$$Q_1 = \mathtt{open}\, n_{\chi_1}.\, R_1 \mid n[\, \overline{\mathtt{open}}\, n_{\chi_2}.\, R_2 \mid R_3\,]$$

$$Q_2 = \mathtt{open}\, m_{\chi_3}.\, S_1 \mid m[\, \overline{\mathtt{open}}\, m_{\chi_4}.\, S_2 \mid S_3\,].$$

We now look at all possible shapes of C_1, C_2, Q_1 and Q_2. Using

$$P^{\star} \equiv C_1[Q_1] \equiv C_2[Q_2]$$

we derive that

$$C_1[Q_1] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[Q_2']$$

$$C_2[Q_2] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[Q_1'].$$

We observe that in $C_1[Q_1] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[Q_2']$ there are three cases: either (i) only Q_1 contributes to the action, or (ii) only C_1 contributes to the action, or (iii) C_1 and Q_1 interact. Similarly in $C_2[Q_2] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[Q_1']$ there are three cases: either (i) only Q_2 contributes to the action, or (ii) only C_2 contributes to the action, or (iii) C_2 and Q_2 interact.
The main point of the proof consists in showing that cases (i) and (ii) cannot occur and that in case (iii) a grave interference pattern occurs. We exploit the fact that there is no process P_3 such that

$$C_1[R_1 \mid R_2 \mid R_3] \longrightarrow_{\{\chi_3,\chi_4\}} P_3$$

$$C_2[S_1 \mid S_2 \mid S_3] \longrightarrow_{\{\chi_1,\chi_2\}} P_3.$$

(i) Assume that only Q_1 contributes to the action in

$$C_1[Q_1] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[Q_2'].$$

We notice that processes R_1 and R_2 cannot contain Q_2 because the capabilities with colors χ_3 and χ_4 are ready to interact in $P^{\star}$. It means that $R_3 \equiv C_3[Q_2]$ for some context C_3 and that

$$C_3[Q_2] \longrightarrow_{\{\chi_3,\chi_4\}} C_3[Q_2'].$$

Therefore, we have

$$C_2[Q_2] \equiv C_1[\mathtt{open}\, n_{\chi_1}.\, R_1 \mid n[\, \overline{\mathtt{open}}\, n_{\chi_2}.\, R_2 \mid C_3[Q_2]\,]]$$

$$C_2[Q_2'] \equiv C_1[\mathtt{open}\, n_{\chi_1}.\, R_1 \mid n[\, \overline{\mathtt{open}}\, n_{\chi_2}.\, R_2 \mid C_3[Q_2']\,]].$$

Arguing by induction on the context, we obtain

$$C_1[Q_1] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[R_1 \mid R_2 \mid C_3[Q_2]] \longrightarrow_{\{\chi_3,\chi_4\}} C_1[R_1 \mid R_2 \mid C_3[Q_2']].$$

Similarly, we obtain also

$$C_2[Q_2] \longrightarrow_{\{\chi_3,\chi_4\}} C_1[\mathtt{open}\, n_{\chi_1} . R_1 \mid n[\,\overline{\mathtt{open}}\, n_{\chi_2} . R_2 \mid C_3[Q_2']\,]]$$

$$C_1[\mathtt{open}\, n_{\chi_1} . R_1 \mid n[\,\overline{\mathtt{open}}\, n_{\chi_2} . R_2 \mid C_3[Q_2']\,]] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[R_1 \mid R_2 \mid C_3[Q_2']].$$

This contradicts, by Lemma A.3, the fact that process $P^\star$ has an interference on $\mathcal{C}_1$ and $\mathcal{C}_2$.
The case when only Q_2 contributes to the action in $C_2[Q_2] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[Q_1']$ is analogous.
(ii) The proof of this case is analogous to that of the corresponding case in (1) (CR-Open,CR-In).
(iii) Assume that C_1 and Q_1 interact in transition

$$C_1[Q_1] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[Q_2']$$

and similarly that C_2 and Q_2 interact in transition

$$C_2[Q_2] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[Q_1'].$$

The only possibility is that ambients n and m coincide and contain both capabilities $\overline{\mathtt{open}}\, n_{\chi_2}$ and $\overline{\mathtt{open}}\, m_{\chi_4}$.
It means that $P^* \equiv C[I]$ where C_{erase} is an active context and I is the following process

$$\mathtt{open}\, n_{\chi_1} . R_1 \mid \mathtt{open}\, n_{\chi_3} . S_1 \mid n[\,\overline{\mathtt{open}}\, n_{\chi_2} . R_2 \mid \overline{\mathtt{open}}\, n_{\chi_4} . S_2 \mid R\,].$$

Process $C[I]_{\mathrm{erase}}$ has a grave interference pattern: case (12) of Definition A.2. By Lemma A.5 we have $P^\star_{\mathrm{erase}} \equiv C[I]_{\mathrm{erase}}$ showing that also P has a plain interference pattern.

(3) (CR-In,CR-Out). Suppose that $Q_1 \longrightarrow_{\mathcal{C}_1} Q_1'$ has been obtained by applying rule CR-Out and that $Q_2 \longrightarrow_{\mathcal{C}_2} Q_2'$ has been obtained by applying rule CR-In. Therefore, we have

$$Q_1 \longrightarrow_{\{\chi_1,\chi_2\}} n[\,R_1 \mid R_2\,] \mid k[\,U_1 \mid U_2\,]$$

$$Q_2 \longrightarrow_{\{\chi_3,\chi_4\}} m[\,h[\,S_1 \mid S_2\,] \mid T_1 \mid T_2\,].$$

where

$$Q_1 = k[\,n[\,\mathtt{out}\, k_{\chi_1} . R_1 \mid R_2\,] \mid \overline{\mathtt{out}}\, k_{\chi_2} . U_1 \mid U_2\,]$$

$$Q_2 = h[\,\mathtt{in}\, m_{\chi_3} . S_1 \mid S_2\,] \mid m[\,\overline{\mathtt{in}}\, m_{\chi_4} . T_1 \mid T_2\,].$$

We now look at all possible shapes of C_1, C_2, Q_1 and Q_2. Using

$$P^\star \equiv C_1[Q_1] \equiv C_2[Q_2]$$

we derive that

$$C_1[Q_1] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[Q_2']$$

$$C_2[Q_2] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[Q_1'].$$

We observe that in $C_1[Q_1] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[Q_2']$ there are three cases: either (i) only Q_1 contributes to the action, or (ii) only C_1 contributes to the action, or (iii) C_1 and Q_1 interact. Similarly, in $C_2[Q_2] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[Q_1']$ there are three cases: either (i) only Q_2 contributes to the action, or (ii) only C_2 contributes to the action, or (iii) C_2 and Q_2 interact.

The main point of the proof consists in showing that cases (i) and (ii) cannot occur and that in case (iii) a grave interference pattern occurs. We exploit the fact that there is no process P_3 such that

$$C_1[n[\,R_1 \mid R_2\,] \mid k[\,U_1 \mid U_2\,]] \longrightarrow_{\{\chi_3,\chi_4\}} P_3$$

$$C_2[m[\,h[\,S_1 \mid S_2\,] \mid T_1 \mid T_2\,]] \longrightarrow_{\{\chi_1,\chi_2\}} P_3.$$

(i) Assume that only Q_1 contributes to the action in

$$C_1[Q_1] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[Q_2].$$

We notice that processes R_1 and U_1 cannot contain Q_2 because the capabilities with colors χ_3 and χ_4 are ready to interact in $P^\star$. Therefore, either R_2 or U_2 contain Q_2. We show only the first case, the other is analogous.

It means that $R_2 \equiv C_3[Q_2]$ for some context C_3 and that

$$C_3[Q_2] \longrightarrow_{\{\chi_3,\chi_4\}} C_3[Q_2'].$$

Therefore, we have

$$C_2[Q_2] \equiv C_1[k[\,n[\,\mathtt{out}\,k_{\chi_1}.\,R_1 \mid C_3[Q_2]\,] \mid \overline{\mathtt{out}}\,k_{\chi_2}.\,U_1 \mid U_2\,]]$$

$$C_2[Q_2'] \equiv C_1[k[\,n[\,\mathtt{out}\,k_{\chi_1}.\,R_1 \mid C_3[Q_2']\,] \mid \overline{\mathtt{out}}\,k_{\chi_2}.\,U_1 \mid U_2\,]].$$

Arguing by induction on the context, we obtain

$$C_1[Q_1] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[n[\,R_1 \mid C_3[Q_2]\,] \mid k[\,U_1 \mid U_2\,]]$$

$$C_1[n[\,R_1 \mid C_3[Q_2]\,] \mid k[\,U_1 \mid U_2\,]] \longrightarrow_{\{\chi_3,\chi_4\}} C_1[n[\,R_1 \mid C_3[Q_2']\,] \mid k[\,U_1 \mid U_2\,]].$$

Similarly, we obtain also

$$C_2[Q_2] \longrightarrow_{\{\chi_3,\chi_4\}} C_1[k[\,n[\,\mathtt{out}\,k_{\chi_1}.\,R_1 \mid C_3[Q_2']\,] \mid \overline{\mathtt{out}}\,k_{\chi_2}.\,U_1 \mid U_2\,]]$$

$$C_1[k[\,n[\,\mathtt{out}\,k_{\chi_1}.\,R_1 \mid C_3[Q_2']\,] \mid \overline{\mathtt{out}}\,k_{\chi_2}.\,U_1 \mid U_2\,]] \longrightarrow_{\{\chi_1,\chi_2\}}$$

$$C_1[n[\,R_1 \mid C_3[Q_2']\,] \mid k[\,U_1 \mid U_2\,]].$$

This contradicts, by Lemma A.3, the fact that process $P^\star$ has an interference on $\mathcal{C}_1$ and $\mathcal{C}_2$.

Assume that only Q_2 contributes to the action in

$$C_2[Q_2] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[Q_1'].$$

We notice that processes S_1 and T_1 cannot contain Q_1 because the capabilities with colors χ_1 and χ_2 are ready to interact in $P^\star$. Therefore, either S_2 or T_2 contain Q_1. We show only the first case, the other is analogous.

It means that $S_2 \equiv C_3[Q_1]$ for some context C_3 and that

$$C_3[Q_1] \longrightarrow_{\{\chi_1,\chi_2\}} C_3[Q_1'].$$

Therefore, we have

$$C_1[Q_1] \equiv C_2[h[\,\mathtt{in}\, m_{\chi_3}.\, S_1 \mid C_3[Q_1]\,] \mid m[\,\overline{\mathtt{in}}\, m_{\chi_4}.\, T_1 \mid T_2\,]]$$

$$C_1[Q_1'] \equiv C_2[h[\,\mathtt{in}\, m_{\chi_3}.\, S_1 \mid C_3[Q_1']\,] \mid m[\,\overline{\mathtt{in}}\, m_{\chi_4}.\, T_1 \mid T_2\,]].$$

Arguing by induction on the context, we obtain

$$C_2[Q_2] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[m[\,h[\,S_1 \mid C_3[Q_1]\,] \mid T_1 \mid T_2\,]]$$

$$C_2[m[\,h[\,S_1 \mid C_3[Q_1]\,] \mid T_1 \mid T_2\,]] \longrightarrow_{\{\chi_1,\chi_2\}} C_2[m[\,h[\,S_1 \mid C_3[Q_1']\,] \mid T_1 \mid T_2\,]].$$

Similarly, we obtain also

$$C_1[Q_1] \longrightarrow_{\{\chi_1,\chi_2\}} C_2[h[\,\mathtt{in}\, m_{\chi_3}.\, S_1 \mid C_3[Q_1']\,] \mid m[\,\overline{\mathtt{in}}\, m_{\chi_4}.\, T_1 \mid T_2\,]]$$

$$C_2[h[\,\mathtt{in}\, m_{\chi_3}.\, S_1 \mid C_3[Q_1']\,] \mid m[\,\overline{\mathtt{in}}\, m_{\chi_4}.\, T_1 \mid T_2\,]] \longrightarrow_{\{\chi_3,\chi_4\}}$$

$$C_2[m[\,h[\,S_1 \mid C_3[Q_1']\,] \mid T_1 \mid T_2\,]].$$

This contradicts, by Lemma A.3, the fact that process $P^\star$ has an interference on $\mathcal{C}_1$ and $\mathcal{C}_2$.

(ii) The proof of this case is analogous to that of the corresponding case in (1) (CR-open,CR-In).

(iii) Assume that C_1 and Q_1 interact in transition

$$C_1[Q_1] \longrightarrow_{\{\chi_3,\chi_4\}} C_2[Q_2']$$

and analogously that C_2 and Q_2 interact in transition

$$C_2[Q_2] \longrightarrow_{\{\chi_1,\chi_2\}} C_1[Q_1'].$$

The only possibility is that one ambient is shared by the two colored reductions. We observe that the case when ambients k and m coincide and contain both capabilities $\overline{\mathtt{out}}\, k_{\chi_2}$ and $\overline{\mathtt{in}}\, m_{\chi_4}$ is not possible. We obtain, in fact, $P_{\text{erase}} \equiv C[R]$ where R is the following process

$$k[\,n[\,\mathtt{out}\, k_{\chi_1}.\, R_1 \mid R_2\,] \mid \overline{\mathtt{out}}\, k_{\chi_2}.\, U_1 \mid \overline{\mathtt{in}}\, k_{\chi_4}.\, T_1 \mid U\,] \mid h[\,\mathtt{in}\, k_{\chi_3}.\, S_1 \mid S_2\,].$$

Process R has the following colored transitions

$$R \longrightarrow_{\{\chi_1,\chi_2\}} n[\,R_1 \mid R_2\,] \mid k[\,\overline{\mathtt{in}}\, k_{\chi_4}.\, T_1 \mid U_1 \mid U\,] \mid h[\,\mathtt{in}\, k_{\chi_3}.\, S_1 \mid S_2\,]$$

$$n[\,R_1 \mid R_2\,] \mid k[\,\overline{\mathtt{in}}\, k_{\chi_4}.\, T_1 \mid U_1 \mid U\,] \mid h[\,\mathtt{in}\, k_{\chi_3}.\, S_1 \mid S_2\,] \longrightarrow_{\{\chi_3,\chi_4\}}$$

$$n[\,R_1 \mid R_2\,] \mid k[\,U_1 \mid T_1 \mid U \mid h[\,S_1 \mid S_2\,]\,]$$

$$R \longrightarrow_{\{\chi_3,\chi_4\}} k[\,n[\,\mathtt{out}\, k_{\chi_1}.\, R_1 \mid R_2\,] \mid \overline{\mathtt{out}}\, k_{\chi_2}.\, U_1 \mid T_1 \mid U \mid h[\,S_1 \mid S_2\,]\,]$$

$$k[\,n[\,\mathtt{out}\, k_{\chi_1}.\, R_1 \mid R_2\,] \mid \overline{\mathtt{out}}\, k_{\chi_2}.\, U_1 \mid T_1 \mid U \mid h[\,S_1 \mid S_2\,]\,] \longrightarrow_{\{\chi_1,\chi_2\}}$$

$$n[\,R_1 \mid R_2\,] \mid k[\,U_1 \mid T_1 \mid U \mid h[\,S_1 \mid S_2\,]\,]\,.$$

This contradicts, by Lemma A.3, the fact that process $P^\star$ has an interference on $\mathcal{C}_1$ and $\mathcal{C}_2$.
Therefore, only the following possibilities remain: either k and h coincide and contain both capabilities $\overline{\mathtt{out}}\, k_{\chi_2}$ and $\mathtt{in}\, m_{\chi_3}$, or n and h coincide and contain both capabilities $\mathtt{in}\, m_{\chi_3}$ and $\mathtt{out}\, k_{\chi_1}$, or m and n coincide and contain both capabilities $\mathtt{out}\, k_{\chi_1}$ and $\overline{\mathtt{in}}\, m_{\chi_4}$.
It means that $P^* \equiv C[I]$ where C_{erase} is an active context and I is one of the following processes
—$k[\, n[\, \mathtt{out}\, k_{\chi_1} . R_1 \mid R_2\,] \mid \overline{\mathtt{out}}\, k_{\chi_2} . U_1 \mid \mathtt{in}\, m_{\chi_3} . S_1 \mid S\,] \mid m[\, \overline{\mathtt{in}}\, m_{\chi_4} . T_1 \mid T_2\,]$;
—$k[\, n[\, \mathtt{out}\, k_{\chi_1} . R_1 \mid \mathtt{in}\, m_{\chi_3} . S_1 \mid R\,] \mid m[\, \overline{\mathtt{in}}\, m_{\chi_4} . T_1 \mid T_2\,] \mid \overline{\mathtt{out}}\, k_{\chi_2} . U_1 \mid U\,]$;
—$k[\, m[\, \mathtt{out}\, k_{\chi_1} . R_1 \mid \overline{\mathtt{in}}\, m_{\chi_4} . T_1 \mid T\,] \mid h[\, \mathtt{in}\, m_{\chi_3} . S_1 \mid S_2\,] \mid \overline{\mathtt{out}}\, k_{\chi_2} . U_1 \mid U\,]$.
In any case, process $C[I]_{\mathrm{erase}}$ has a grave interference pattern: cases (8), (6) and (7) of Definition A.2, respectively. By Lemma A.5 we have $P^\star_{\mathrm{erase}} \equiv C[I]_{\mathrm{erase}}$ showing that also P has a plain interference pattern.

(4) (CR-Open,CR-Out). The proof is similar to that of case (1) (CR-Open, CR-In).
(5) (CR-In,CR-In). The proof is similar to that of cases (3) (CR-Open,CR-Open) and (4) (CR-In, CR-Out).
(6) (CR-Out,CR-Out). The proof is similar to that of cases (3) (CR-Open, CR-Open) and (4) (CR-In,CR-Out).

□

Using Definitions A.2 and A.1 and the result on types in Section 5 we can now give the proof of Corollary 5.14, which shows that ST and immobility types eliminate the grave interferences. For convenience of the reader we recall its assertion:

> *Suppose* $\Gamma \vdash P\ :\ \mathit{Proc}$ *, and* P *is closed. Suppose also that the type of each name defined in* Γ *and each restricted name in* P *is immobile or ST. For all* P' *such that* $P \Longrightarrow P'$*, no grave interferences occur in* P'*.*

Proof. **(of Corollary 5.14)** By induction on the length of $\Longrightarrow$. For the base case, we use Theorems 5.11 and 5.12, Definition A.2 and Lemma A.6. For the inductive case, we use Theorem 5.10. □

B. A LABELED TRANSITION SEMANTICS

We present a labeled transition semantics of SA, which we use to prove the laws of Section 7. Transitions are of the form $P \xrightarrow{\alpha} O$, where label α and outcome O are defined by the following grammar:

μ	$:= \mathtt{in}\, n \mid \mathtt{out}\, n \mid \overline{\mathtt{in}}\, n \mid \overline{\mathtt{out}}\, n \mid \mathtt{open}\, n \mid \overline{\mathtt{open}}\, n$	actions and coactions
α	$:= \mu \mid \tau \mid (M) \mid \langle - \rangle \mid ?n \mid ?\mathtt{open}\, n$	Labels
K	$:= \boldsymbol{\nu} \tilde{n}\, \langle P \rangle Q$	concretions
O	$:= P \mid K$	outcomes

Labels (M) and $\langle - \rangle$ are used for input and output transitions. Outcomes are processes and concretions. A transition $P \xrightarrow{\alpha} O$ for $\alpha \neq \tau$ is a *visible* transition.

The rules for the labeled transitions are shown in Tables XIV and XV. If O is the concretion $\boldsymbol{\nu}\tilde{m}\,\langle P\rangle R$, then:

—$\boldsymbol{\nu}nO$ is the concretion $\boldsymbol{\nu}\tilde{m}\langle P\rangle(\boldsymbol{\nu}nR)$, if $n \notin \mathrm{fn}(P)$, and the concretion $\boldsymbol{\nu}(n, \tilde{m})\langle P\rangle R$ otherwise;

—$O \mid Q$ is the concretion $\boldsymbol{\nu}\tilde{m}\,\langle P\rangle(R \mid Q)$, provided $\mathrm{fn}(Q) \cap \{\tilde{m}\} = \emptyset$.

Visible transitions, defined in Table XIV, are complex because it is necessary to represent two different situations:

(1) an ambient can interact, by moving into or out of another ambient, accepting to be opened, or accepting movements into and out of itself;

(2) a process can exercise a capability.

Whenever a process P exercising a capability is placed inside an ambient n, we obtain a transition representing a request of interaction from ambient n (rules co-open, co-in, in-out). In particular:

—If P reduces to Q by exercising a coaction $\overline{\texttt{open}}\,n$, then ambient n is ready to be opened. This is represented by a transition $n[\,P\,] \xrightarrow{?\texttt{open}\,n} Q$, in rule co-open.

—If P reduces to Q by exercising an action $\texttt{in}\,m$, then ambient n can move inside m. This is represented by a transition $n[\,P\,] \xrightarrow{\texttt{in}\,m} \langle n[\,Q\,]\rangle\mathbf{0}$, in rule in-out. (The case of a transition labeled $\texttt{out}\,m$ is similar.)

—If P reduces to Q by exercising a coaction $\overline{\texttt{in}}\,n$, then an ambient n in P allows another ambient to enter. This is represented by a transition $n[\,P\,] \xrightarrow{\overline{\texttt{in}}\,n} \langle Q\rangle\mathbf{0}$, in rule co-in.

In all these cases, process Q is the process contained in ambient n after the execution of $\overline{\texttt{in}}\,n$ or $\texttt{in}\,m$ or $\texttt{out}\,m$, and $\mathbf{0}$ is the process which remains outside. The process outside can however grow by means of rule par, for instance.

Leaving aside rule co-out for now, we explain τ-transitions, in Table XV. These are obtained by combining transitions representing requests of interaction from ambients and capabilities exercised by processes. Rule tau-in realizes the synchronization between a process in which an ambient wants to enter inside n and a process in which an ambient n allows ambients to enter. In the rule, the processes in concretions $\boldsymbol{\nu}\tilde{p}\,\langle P_1\rangle P_2$ and $\boldsymbol{\nu}\tilde{q}\,\langle Q_1\rangle Q_2$ have the following meaning: P_1 is the ambient which wants to enter inside ambient n, and P_2 is its residual process (which does not move); Q_1 is the remaining process inside n after the execution of $\overline{\texttt{in}}\,n$, and Q_2 is its residual process (which does not move). Rule tau-open realizes the synchronization between a process exercising a capability $\texttt{open}\,n$ and a process in which an ambient n is ready to be opened.

We now explain co-out. An ambient that wants to move out of n needs the permission from a parallel process through the capability $\overline{\texttt{out}}\,n$. In the rule, this is represented by a transition with label $?n$ and with target the concretion $\boldsymbol{\nu}\tilde{n}\langle P_1\rangle(P_2 \mid R)$, where P_1 is the ambient which is authorised to move out of n, and $P_2 \mid R$ is the residual process (which does not move from n). Rule tau-out realizes the synchronization between a process in which an ambient has the permission to go out of n and ambient n.

Table XIV. Rules for visible transitions

$$\mu.\,P \xrightarrow{\mu} P \quad \text{cap} \qquad (x)\,P \xrightarrow{(M)} P\{M/x\} \quad \text{com}$$

$$\langle M\rangle \xrightarrow{\langle - \rangle} \langle\langle M\rangle\rangle\mathbf{0} \quad \text{Msg} \qquad \frac{M.\,(M'.\,P) \xrightarrow{\alpha} Q}{(M.\,M').\,P \xrightarrow{\alpha} Q} \quad \text{path}$$

$$\frac{P \xrightarrow{\overline{\texttt{open}}\,m} Q}{m[\,P\,] \xrightarrow{?\texttt{open}\,m} Q} \quad \text{co-open} \qquad \frac{P \xrightarrow{\alpha} O}{P \mid Q \xrightarrow{\alpha} O \mid Q} \quad \text{par}$$

$$\frac{P \xrightarrow{\mu} Q \qquad \mu \in \{\texttt{in}\,m, \texttt{out}\,m\}}{n[\,P\,] \xrightarrow{\mu} \langle n[\,Q\,]\rangle\mathbf{0}} \quad \text{in-out} \qquad \frac{P \xrightarrow{\overline{\texttt{in}}\,n} Q}{n[\,P\,] \xrightarrow{\overline{\texttt{in}}\,n} \langle Q\rangle\mathbf{0}} \quad \text{co-in}$$

$$\frac{P \xrightarrow{\alpha} O}{\boldsymbol{\nu} n\ P \xrightarrow{\alpha} \boldsymbol{\nu} n\ O} \quad (\text{if } n \text{ not free in } \alpha) \quad \text{res}$$

$$\frac{P \xrightarrow{\texttt{out}\,n} \boldsymbol{\nu}\tilde{n}\,\langle P_1\rangle P_2 \qquad Q \xrightarrow{\overline{\texttt{out}}\,n} R}{P \mid Q \xrightarrow{?n} \boldsymbol{\nu}\tilde{n}\,\langle P_1\rangle(P_2 \mid R)} \quad (\text{if } \mathrm{fn}(R) \cap \{\tilde{n}\} = \emptyset) \quad \text{co-out}$$

Table XV. Rules for τ transitions

$$\epsilon.\,P \xrightarrow{\tau} P \quad \text{tau-eps}$$

$$\frac{P \xrightarrow{\tau} Q}{m[\,P\,] \xrightarrow{\tau} m[\,Q\,]} \quad \text{tau-amb} \qquad \frac{P \xrightarrow{\texttt{open}\,n} P' \qquad Q \xrightarrow{?\texttt{open}\,n} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \quad \text{tau-open}$$

$$\frac{P \xrightarrow{\texttt{in}\,n} \boldsymbol{\nu}\tilde{p}\,\langle P_1\rangle P_2 \qquad Q \xrightarrow{\overline{\texttt{in}}\,n} \boldsymbol{\nu}\tilde{q}\,\langle Q_1\rangle Q_2}{P \mid Q \xrightarrow{\tau} \boldsymbol{\nu}\tilde{p}\,\boldsymbol{\nu}\tilde{q}\,(n[\,P_1 \mid Q_1\,] \mid (P_2 \mid Q_2))} \quad \begin{array}{l}(\text{ if } (\mathrm{fn}(Q_1) \cup \mathrm{fn}(Q_2)) \cap \{\tilde{p}\} = \emptyset\ )\\ (\mathrm{fn}(P_1) \cup \mathrm{fn}(P_2)) \cap \{\tilde{q}\} = \emptyset\end{array} \quad \text{tau-in}$$

$$\frac{P \xrightarrow{\langle - \rangle} \boldsymbol{\nu}\tilde{n}\,\langle\langle M\rangle\rangle P_1 \qquad Q \xrightarrow{(M)} Q_1}{P \mid Q \xrightarrow{\tau} \boldsymbol{\nu}\tilde{n}\,(P_1 \mid Q_1)} \quad (\text{if } \mathrm{fn}(Q_1) \cap \{\tilde{n}\} = \emptyset) \quad \text{tau-com}$$

$$\frac{P \xrightarrow{?n} \boldsymbol{\nu}\tilde{n}\,\langle P_1\rangle P_2}{n[\,P\,] \xrightarrow{\tau} \boldsymbol{\nu}\tilde{n}\,(P_1 \mid n[\,P_2\,])} \quad \text{tau-out} \qquad \texttt{rec}\,X.\,P \xrightarrow{\tau} P\{\texttt{rec}\,X.\,P/X\} \quad \text{tau-rec}$$

It will be useful for later proofs to extend the labeled transitions also to contexts. These transitions are defined as for processes—there is no rule for the hole. Similarly we extend the structural congruence relation $\equiv$ to contexts.

The theorem below asserts the agreement between the labeled transition semantics and the reduction semantics.

Theorem B.1.

—*If* $P \xrightarrow{\tau} P'$ *then* $P \longrightarrow P'$

—*If* $P \longrightarrow P'$ *then* $P \xrightarrow{\tau}\equiv P'$

PROOF. This is a standard result for reduction semantics for calculi of mobile processes such as the π-calculus [Sangiorgi and Walker 2001, Lemma 1.4.15(2)]. The proof for SA is similar. $\square$

COROLLARY B.2. **(subject reduction, using the labeled transitions)** *If* $\Gamma \vdash P : Proc$ *and* $P \xrightarrow{\tau} P'$ *then also* $\Gamma \vdash P' : Proc$ *.*

PROOF. From Theorems 5.10 and B.1. $\square$

The observable predicates $P \Downarrow_n$ (Section 6) can also be defined in terms of labeled transitions. We write $\xrightarrow{\tau}_\star$ for the reflexive and transitive closure of $\xrightarrow{\tau}$:

PROPOSITION B.3. *We have* $P \Downarrow_n$ *if and only if* $P \xrightarrow{\tau}_\star P'$, *for some* P' *such that either there exists a concretion* K *such that* $P' \xrightarrow{\overline{\mathtt{in}}\, n} K$ *or there exists a process* Q *such that* $P' \xrightarrow{?\mathtt{open}\, n} Q$.

C. PROOFS OF SECTION 7

In the light of Theorem B.1 and Proposition B.3, in the definition of barbed bisimulation (Definition 6.1) we can use the labeled transitions in place of the reduction relation. We will also use the standard "up to technique" below.

Definition C.1. **(barbed bisimulation up to $\equiv$)** A symmetric relation $\mathcal{R}$ on closed processes is a *barbed bisimulation up to* $\equiv$ if $P \mathrel{\mathcal{R}} Q$ implies:

(1) whenever $P \xrightarrow{\tau}_\star P'$ then there exists Q' such that $Q \xrightarrow{\tau}_\star Q'$ and $P' \equiv\mathcal{R}\equiv Q'$;
(2) for each name n, $P \Downarrow_n$ iff $Q \Downarrow_n$.

LEMMA C.2. *If* $\mathcal{R}$ *is a barbed bisimulation up to* $\equiv$, *then* $\mathcal{R} \subseteq \dot{\simeq}$.

We sometimes use the above technique without mentioning it. We write $\mathcal{I}$ for the identity relation, $P \not\downarrow_h$ if not $P \downarrow_h$, and $P \not\xrightarrow{\alpha}$ if there is no O such that $P \xrightarrow{\alpha} O$.

Untyped calculus. Cases (1-6) of Lemma 7.1 follow from Lemma C.3; case (7) is trivial.

LEMMA C.3. *Let* $\mathcal{R}_i^{k_c} = \{(C[R_{1,i,1},\ldots,R_{1,i,k_c}], C[R_{2,i,1},\ldots,R_{2,i,k_c}])\}$ *be the process relations such that* C *is a polyadic context (of arity* k_c*) and for* $i \in \{1..6\}$ *and* $j \in \{1..k_c\}$*:*

(1) $R_{1,1,j} = \boldsymbol{\nu} n\, (m[\,\mathtt{in}\, n.\, P_j\,] \mid n[\,\overline{\mathtt{in}}\, n.\, Q_j\,])$ *and* $R_{2,1,j} = \boldsymbol{\nu} n\, (n[\,Q_j \mid m[\,P_j\,]\,])$;
(2) $R_{1,2,j} = k[\,m[\,\mathtt{in}\, n.\, P_j\,] \mid n[\,\overline{\mathtt{in}}\, n.\, Q_j\,] \mid \overline{\mathtt{out}}\, k.\, R\,]$ *and* $R_{2,2,j} = k[\,n[\,Q_j \mid m[\,P_j\,]\,] \mid \overline{\mathtt{out}}\, k.\, R\,]$;
(3) $R_{1,3,j} = \boldsymbol{\nu} m\, (\mathtt{open}\, m.\, P_j \mid m[\,\overline{\mathtt{open}}\, m.\, Q_j\,])$ *and* $R_{2,3,j} = \boldsymbol{\nu} m\, (P_j \mid Q_j)$;
(4) $R_{1,4,j} = k[\,\mathtt{open}\, m.\, P_j \mid m[\,\overline{\mathtt{open}}\, m.\, Q_j\,]\,]$ *and* $R_{2,4,j} = k[\,P_j \mid Q_j\,]$;
(5) $R_{1,5,j} = n[\,\overline{\mathtt{out}}\, n.\, P_j \mid m[\,\mathtt{out}\, n.\, Q_j\,]\,]$ *and* $R_{2,5,j} = m[\,Q_j\,] \mid n[\,P_j\,]$;
(6) $R_{1,6,j} = n[\,\langle M\rangle \mid (x)\, P_j\,]$ *and* $R_{2,6,j} = n[\,P_j\{M/x\}\,]$.

For every $i \in \{1..6\}$ *relation* $\mathcal{I} \cup \bigcup_{k_c} \mathcal{R}_i^{k_c}$ *is a weak barbed bisimulation.*

PROOF. For simplicity we show the proof for the case $k_c = 1$; we abbreviate $R_{1,i,1}$ as $R_{1,i}$, $R_{2,i,1}$ as $R_{2,i}$, and P_1, Q_1 as P, Q, and relation $\mathcal{R}_i^1$ as $\mathcal{R}_i$.

Let $C[R_{1,i}]\ \mathcal{R}_i\ C[R_{2,i}]$ for some $i \in \{1..6\}$. It holds that $R_{1,i} \xrightarrow{\alpha} O$ iff $\alpha = \tau$ and $O = R_{2,i}$. Hence, the case where $C[R_{2,i}]$ moves and $C[R_{1,i}]$ has to answer is straightforward. For instance, if $C[R_{2,i}] \xrightarrow{\tau} R'$ and $R_{2,i}$ contributes to the action, then also $C[R_{1,i}] \xrightarrow{\tau}_\star R'$.

We consider now the converse, when $C[R_{1,i}]$ moves. Since $R_{1,i} \not\downarrow_h$, for every name h, any observable $C[R_{1,i}] \downarrow_h$ is only caused by the context C, therefore also $C[R_{2,i}] \downarrow_h$. We now look at the matching on τ-transitions. Since $R_{1,i} \not\xrightarrow{\alpha}$ for every label $\alpha \neq \tau$, no interaction between $R_{1,i}$ and the context is possible. In conclusion, we have $C[R_{1,i}] \xrightarrow{\tau} R'$ in one of the following cases:

(1) $R_{1,i}$ contributes to the action. Since $R_{1,i} \xrightarrow{\alpha} O$ iff $\alpha = \tau$ and $O = R_{2,i}$, we have $R' = C[R_{2,i}]$. In this case the other process, $C[R_{2,i}]$ need not move, and we are done because of the identity relation $\mathcal{I}$.

(2) $R_{1,i}$ does not contribute to the action. By induction on the depth of the inference the last rule used may have been tau-open, tau-out, tau-in, tau-com, tau-amb, res, par, rec. We show some cases only, the others are similar.

(a) If $C[R_{1,i}] \xrightarrow{\tau} R'$ has been obtained by rule tau-open, then $C[R_{1,i}] = P' \mid Q'$ with $P' \xrightarrow{\mathsf{open}\,k} P''$, $Q' \xrightarrow{?\mathsf{open}\,k} Q''$ and $R' = P'' \mid Q''$. There are two cases: either the hole occurs on the left (that is, $P' = C_1[R_{1,i}]$) or the hole occurs on the right (that is, $Q' = C_1[R_{1,i}]$).

Assume $P' = C_1[R_{1,i}]$. Since $R_{1,i} \not\xrightarrow{\mathsf{open}\,k}$, then (arguing by induction on the depth of the inference) $P'' = C_2[R_{1,i}]$ with $C_1 \xrightarrow{\mathsf{open}\,k} C_2$. Hence, we also have

$$C_1[R_{2,i}] \xrightarrow{\mathsf{open}\,k} C_2[R_{2,i}].$$

Thus, by rule tau-open, using as premises $C_1[R_{2,i}] \xrightarrow{\mathsf{open}\,k} C_2[R_{2,i}]$ and $Q' \xrightarrow{?\mathsf{open}\,k} Q''$, we obtain

$$C[R_{2,i}] \xrightarrow{\tau} C_2[R_{2,i}] \mid Q''.$$

Transition $C[R_{2,i}] \xrightarrow{\tau} C_2[R_{2,i}] \mid Q''$ matches transition $C[R_{1,i}] \xrightarrow{\tau} C_2[R_{1,i}] \mid Q''$, because $C_2[R_{1,i}] \mid Q''\ \mathcal{R}_i\ C_2[R_{2,i}] \mid Q''$. The remaining case is similar.

(b) If transition $C[R_{1,i}] \xrightarrow{\tau} R'$ has been obtained by rule tau-com, then $C[R_{1,i}] = P' \mid Q'$ with $P' \xrightarrow{\langle - \rangle} \boldsymbol{\nu}\tilde{n}\ \langle\langle M \rangle\rangle P''$, and $Q' \xrightarrow{(M)} Q''$, and $R' = \boldsymbol{\nu}\tilde{n}\ (P'' \mid Q'')$. There are two cases: either the hole occurs on the left (that is, $P' = C_1[R_{1,i}]$), or the hole occurs on the right (that is, $Q' = C_1[R_{1,i}]$).

—Assume $P' = C_1[R_{1,i}]$. Since $R_{1,i} \not\xrightarrow{\langle - \rangle}$, then (arguing by induction on the depth of the inference) $P'' = C_2[R_{1,i}]$ with $C_1 \xrightarrow{\langle - \rangle} \boldsymbol{\nu}\tilde{n}\ \langle\langle M \rangle\rangle C_2$. Hence, we also have

$$C_1[R_{2,i}] \xrightarrow{\langle - \rangle} \boldsymbol{\nu}\tilde{n}\ \langle\langle - \rangle\rangle C_2[R_{2,i}].$$

By rule tau-com, using as premises $C_1[R_{2,i}] \xrightarrow{\langle - \rangle} \boldsymbol{\nu}\tilde{n}\,\langle\langle M\rangle\rangle C_2[R_{2,i}]$ and $Q' \xrightarrow{(M)} Q''$, we obtain

$$C[R_{2,i}] \xrightarrow{\tau} \boldsymbol{\nu}\tilde{n}\,(C_2[R_{2,i}] \mid Q'').$$

Transition $C[R_{2,i}] \xrightarrow{\tau} \boldsymbol{\nu}\tilde{n}\,(C_2[R_{2,i}] \mid Q'')$ matches transition $C[R_{1,i}] \xrightarrow{\tau} \boldsymbol{\nu}\tilde{n}\,(C_2[R_{1,i}] \mid Q'')$, because $\boldsymbol{\nu}\tilde{n}\,(C_2[R_{1,i}] \mid Q'')\ \mathcal{R}_i\ \boldsymbol{\nu}\tilde{n}\,(C_2[R_{2,i}] \mid Q'')$.

—Assume $Q' = C_1[R_{1,i}]$. Arguing by induction on the depth of the inference, we find that there are two cases, depending on whether $R_{1,i}$ takes part in the communication: either $Q'' = C_2[R_{1,i}\{^M\!/\!_x\}]$, or $Q'' = C_2[R_{1,i}]$ and $C_1 \xrightarrow{(M)} C_2$. The two cases are similar, we only consider the first. Since $C_1[R_{1,i}] \xrightarrow{(M)} C_2[R_{1,i}\{^M\!/\!_x\}]$, then we also have (by induction on the depth of the inference)

$$C_1[R_{2,i}] \xrightarrow{(M)} C_2[R_{2,i}\{^M\!/\!_x\}].$$

Hence, by rule tau-com, using as premises $P' \xrightarrow{\langle - \rangle} \boldsymbol{\nu}\tilde{n}\,\langle\langle M\rangle\rangle P''$ and $C_1[R_{2,i}] \xrightarrow{(M)} C_2[R_{2,i}\{^M\!/\!_x\}]$, we obtain

$$C[R_{2,i}] \xrightarrow{\tau} \boldsymbol{\nu}\tilde{n}\,(P'' \mid C_2[R_{2,i}\{^M\!/\!_x\}]).$$

Transition $C[R_{2,i}] \xrightarrow{\tau} \boldsymbol{\nu}\tilde{n}(P'' \mid C_2[R_{2,i}\{^M\!/\!_x\}])$ matches transition $C[R_{1,i}] \xrightarrow{\tau} \boldsymbol{\nu}\tilde{n}\,(P'' \mid C_2[R_{1,i}\{^M\!/\!_x\}])$, because $\boldsymbol{\nu}\tilde{n}\,(P'' \mid C_2[R_{1,i}\{^M\!/\!_x\}])\ \mathcal{R}_i\ \boldsymbol{\nu}\tilde{n}\,(P'' \mid C_2[R_{2,i}\{^M\!/\!_x\}])$.

(c) Assume that transition $C[R_{1,i}] \xrightarrow{\tau} R'$ has been obtained by rule rec. It means that $C = \mathtt{rec}\,X.\,C_1$ for some monadic context C_1 and that $R' = C_1[R_{1,i}]\{^{\mathtt{rec}\,X.\,C_1[R_{1,i}]}\!/\!_X\}$.
By rule rec we also have

$$\mathtt{rec}\,X.\,C_1[R_{2,i}] \xrightarrow{\tau} C_1[R_{2,i}]\{^{\mathtt{rec}\,X.\,C_1[R_{2,i}]}\!/\!_X\}.$$

Let D be the polyadic context $C_1\{^{\mathtt{rec}\,X.\,C_1}\!/\!_X\}$, of some arity k_c. Transition $C[R_{2,i}] \xrightarrow{\tau} D[R_{2,i},\ldots,R_{2,i}]$ matches transition $C[R_{1,i}] \xrightarrow{\tau} D[R_{1,i},\ldots,R_{1,i}]$, because $D[R_{1,i},\ldots,R_{1,i}]$
$\mathcal{R}_i^{k_c}\ D[R_{2,i},\ldots,R_{2,i}]$.

The proof for polyadic contexts is similar: the processes placed in the holes cannot synchronize with the environment or with each other, since, for every $i \in \{1..6\}$, and $j \in \{1..k_c\}$, and $\alpha \neq \tau$, it holds that $R_{1,i,j} \not\xrightarrow{\alpha}$. □

Typed calculi. In order to prove the typed laws of Lemma 7.2 we exploit subject reduction (in terms of labeled transitions, as by Corollary B.2), and the other properties of types shown in Section 5 (which by Theorem B.1 also holds for the labeled transitions). Lemma 7.2 follows from Lemma C.6 and Lemma C.7. The proof of Corollary 7.4 is analogous to those of Lemma 7.1 and Lemma 7.2.

LEMMA C.4. *Let Q be a process which doesn't have unguarded capabilities or messages or abstractions. If $Q \xrightarrow{\tau} Q'$, then Q' does not have unguarded abstractions.*

PROOF. Suppose that there exists Q' with an unguarded abstraction. There are two possibilities: either a top-level ambient has been opened or a capability has been consumed. The latter case is impossible since Q doesn't have unguarded capabilities or messages. The first case requires the existence of a capability $\mathtt{open}\, m$ unguarded in Q.

□

PROPOSITION C.5. *Let C be a context that respects the linearity of L. If $C \xrightarrow{\tau} D$, then also D respects the linearity of L.*

LEMMA C.6. *Let $\mathcal{R}_i = \{(C[R_{1,i}], C[R_{2,i}])\}$ be the process relations such that, for $i \in \{1, 2\}$:*

—$\Gamma \vdash C[R_{1,i}]$: *Proc and ambients n and m are ST in Γ;*

—*C is a context that respects the linearity of L_i, for $L_1 = \{\mathtt{in}\, n, \overline{\mathtt{in}}\, n\}$ and $L_2 = \{\mathtt{open}\, m, \overline{\mathtt{open}}\, m\}$;*

—*processes $R_{1,i}$ and $R_{2,i}$ are defined as follows:*

(1) $R_{1,1} = m[\, \mathtt{in}\, n. P_1 \mid P_2\,] \mid n[\,\overline{\mathtt{in}}\, n. Q_1 \mid Q_2\,]$ *and* $R_{2,1} = n[\, Q_1 \mid Q_2 \mid m[\, P_1 \mid P_2\,]\,]$;

(2) $R_{1,2} = \mathtt{open}\, m. P \mid m[\,\overline{\mathtt{open}}\, m. Q_1 \mid Q_2\,]$ *and* $R_{2,2} = P \mid Q_1 \mid Q_2$.

For each $i \in \{1, 2\}$, relation $\mathcal{I} \cup \mathcal{R}_i$ is a weak barbed bisimulation with respect to L_i.

PROOF. The proof for $i = 1$ is similar to that of case (1) of Lemma C.3, and the proof for $i = 2$ is similar to that of case (3) of Lemma C.3. The main observation is that both $R_{1,1}$ and $R_{1,2}$ cannot interact with the context: plain interferences are avoided by the constraint on the use of capabilities (rather than by restriction, as in Lemma C.3); grave interferences are controlled by types (rather than by the uniqueness of local processes, as in Lemma C.3).

Let $C[R_{1,i}]\ \mathcal{R}_i\ C[R_{2,i}]$ for some $i \in \{1, 2\}$. The case where $C[R_{2,i}]$ moves, and $C[R_{1,i}]$ has to answer, is immediate, because $R_{1,i} \xrightarrow{\tau} R_{2,i}$. Thus, if $C[R_{2,i}] \xrightarrow{\tau} R'$ and $R_{2,i}$ contributes to the action, then also $C[R_{1,i}] \xrightarrow{\tau}_{\star} R'$.

We consider now the converse, when $C[R_{1,i}]$ moves, for $i \in \{1, 2\}$.

(1) Suppose $C[R_{1,1}]$ moves. Since m and n are ST, then (by Theorem 5.12) P_2 and Q_2 cannot exercise any capability. Hence, we have $R_{1,1} \xrightarrow{\alpha} O$ with $\alpha \neq \tau$ iff $\alpha \in \{\mathtt{in}\, n, \overline{\mathtt{in}}\, n\}$. Since $L_1 = \{\mathtt{in}\, n, \overline{\mathtt{in}}\, n\}$, capabilities $\mathtt{in}\, n$ and $\overline{\mathtt{in}}\, n$ are not observable; that is, $R_{1,1} \not\downarrow_h^{L_1}$ for every name h. Therefore, any observable $C[R_{1,1}] \downarrow_h^{L_1}$ is only caused by the context C, so that also $C[R_{2,1}] \downarrow_h^{L_1}$ holds.
We look at the matching of τ-transitions. There are two cases depending on whether $R_{1,1}$ contributes to the action. The most interesting case is when $R_{1,1}$ contributes to the action. In the other case the proof is similar to the corresponding case in proof of Lemma C.3 with a minor modification: monadic contexts are enough to establish a barbed bisimulation. Since contexts respects the linearity of L_i, by Definition 6.4, the hole cannot occur under recursion.
Suppose $R_{1,1}$ contributes to the action. Since C respects the linearity of L_1, no ambient (of the context) can enter inside n, and ambient m cannot move inside any ambient n (of the context). Furthermore, since $R_{1,1} \xrightarrow{\alpha} O$ with $\alpha \neq \tau$ iff $\alpha \in \{\mathtt{in}\, n, \overline{\mathtt{in}}\, n\}$, no ambient (of the context) can enter inside m, no ambient

can exit from n and m, ambients n and m cannot be opened, and n cannot move. In conclusion, no interaction between $R_{1,1}$ and the context is possible.
Hence, $R' = C[R'']$ with $R_{1,1} \xrightarrow{\tau} R''$; that is, either $R'' = R_{2,1}$, or $R'' = m[\,\mathtt{in}\, n.\, P_1 \mid P_2'\,] \mid n[\,\overline{\mathtt{in}}\, n.\, Q_1 \mid Q_2\,]$ with $P_2 \xrightarrow{\tau} P_2'$, or $R'' = m[\,\mathtt{in}\, n.\, P_1 \mid P_2\,] \mid n[\,\overline{\mathtt{in}}\, n.\, Q_1 \mid Q_2'\,]$ with $Q_2 \xrightarrow{\tau} Q_2'$.
When $R'' = R_{2,1}$ we are done because of the identity relation $\mathcal{I}$. Assume $R'' = m[\,\mathtt{in}\, n.\, P_1 \mid P_2'\,] \mid n[\,\overline{\mathtt{in}}\, n.\, Q_1 \mid Q_2\,]$. We therefore have $R_{2,1} \xrightarrow{\tau} R'''$, where $R''' = n[\,Q_1 \mid Q_2 \mid m[\,P_1 \mid P_2'\,]\,]$, and also $C[R_{2,1}] \xrightarrow{\tau} C[R''']$. We conclude that $C[R'']\ \mathcal{R}_1\ C[R''']$, because by subject reduction both $\Gamma \vdash C[R''] : \mathit{Proc}$ and $\Gamma \vdash C[R'''] : \mathit{Proc}$ hold. The remaining case is similar.

(2) Suppose $C[R_{1,2}]$ moves. Since m is ST, (by Theorem 5.12) Q_2 cannot exercise any capability. Hence, we have $R_{1,2} \xrightarrow{\alpha} O$ with $\alpha \neq \tau$ if and only if either $\alpha = \mathtt{open}\, m$ and $O = P \mid m[\,\overline{\mathtt{open}}\, m.\, Q_1 \mid Q_2\,]$, or $\alpha = ?\mathtt{open}\, m$ and $O = \mathtt{open}\, m.\, P \mid Q_1 \mid Q_2$. Since $L_2 = \{\mathtt{open}\, m, \overline{\mathtt{open}}\, m\}$, capabilities $\mathtt{open}\, m$ and $?\mathtt{open}\, m$ are not observable. Therefore, any observable $C[R_{1,2}] \downarrow_h^{L_2}$ is caused by the context C, so that also $C[R_{2,2}] \downarrow_h^{L_2}$ holds.
We now look at the matching of τ-transitions. There are two cases depending on whether $R_{1,2}$ contributes to the action. The most interesting case is when $R_{1,2}$ contributes to the action. In the other case the proof is similar to the corresponding case in the proof of Lemma C.3.
Suppose $R_{1,2}$ contributes to the action. Since C respects the linearity of L_2, no process (of the context) can open m, and no ambient m (of the context) can be opened. Furthermore, by Theorem 5.12 no ambient (of the context) can enter inside m, no ambient can exit from m, and m cannot move. We conclude that no interaction is possible between $R_{1,2}$ and the context.
Hence, $R' = C[R'']$ with $R_{1,2} \xrightarrow{\tau} R''$; that is, either $R'' = R_{2,2}$, or $R'' = \mathtt{open}\, m.\, P \mid m[\,\overline{\mathtt{open}}\, m.\, Q_1 \mid Q_2'\,]$ with $Q_2 \xrightarrow{\tau} Q_2'$.
The former case is immediate because of the identity relation $\mathcal{I}$. In the latter case, we have $R_{2,2} \xrightarrow{\tau} R'''$, where $R''' = P \mid Q_1 \mid Q_2'$, and also $C[R_{2,2}] \xrightarrow{\tau} C[R''']$. We conclude that $C[R'']\ \mathcal{R}_2\ C[R''']$, because by subject reduction both $\Gamma \vdash C[R''] : \mathit{Proc}$ and $\Gamma \vdash C[R'''] : \mathit{Proc}$ hold.

□

LEMMA C.7. *Let* $\mathcal{R}_i^{k_c} = \{(C[R_{1,i,1}, \ldots, R_{1,i,k_c}], C[R_{2,i,1}, \ldots, R_{2,i,k_c}])\}$ *be the process relations, where* C *is a polyadic context such that* $\Gamma \vdash C[R_{1,i,1}, \ldots, R_{1,i,k_c}] : \mathit{Proc}$, *for* $i \in \{1,2\}$, *ambients* n *and* m *are* ST *in* Γ, *and for* $j \in \{1..k_c\}$*:*

(1) $R_{1,1,j} = n[\,\overline{\mathtt{out}}\, n.\, P_{1,j} \mid m[\,\mathtt{out}\, n.\, Q_{1,j} \mid Q_{2,j}\,] \mid P_{2,j}\,]$ *and* $R_{2,1,j} = m[\,Q_{1,j} \mid Q_{2,j}\,] \mid n[\,P_{1,j} \mid P_{2,j}\,]$ *with* $P_{2,j} \not\stackrel{\mathtt{out}\, n}{\Longrightarrow}$ *;*

(2) $R_{1,2,j} = n[\,\langle M \rangle \mid (x)\, P_j \mid Q_j\,]$ *and* $R_{2,2,j} = m[\,P_j\{M/x\} \mid Q_j\,]$ *where the messages in* n *carry the thread (i.e.,* $\Gamma(n)$ *is of the form* $\mathtt{STAmb}\, I\, [\uparrow W]$*), and no abstractions appear unguarded in* Q_j*.*

For $i \in \{1,2\}$, *relation* $\mathcal{I} \cup \bigcup_{k_c} \mathcal{R}_i^{k_c}$ *is a weak barbed bisimulation.*

PROOF. The proof for $i = 1$ is similar to that of case (5) of Lemma C.3 and the proof for $i = 2$ is similar to that of case (6) of Lemma C.3. The key point is that neither $R_{1,1,j}$ nor $R_{1,2,j}$ can interact with the context: as in Lemma C.3 plain interferences are avoided by enclosing the interacting processes inside ambient n, while grave interferences are removed by the types.

For simplicity, we show the case of $k_c = 1$. We abbreviate $R_{1,i,1}$ as $R_{1,i}$, and $R_{2,i,1}$ as $R_{2,i}$, and $P_{i,j}$, $Q_{i,j}$ as P_i and Q_i, and P_j, Q_j as P and Q. Moreover, we abbreviate relation $\mathcal{R}_i^1$ as $\mathcal{R}_i$.

Let $C[R_{1,i}]\ \mathcal{R}_i\ C[R_{2,i}]$ for some $i \in \{1,2\}$. The case where $C[R_{2,i}]$ moves, and $C[R_{1,i}]$ has to answer, is easy, because $R_{1,i} \xrightarrow{\tau} R_{2,i}$.
Below we consider the converse, when $C[R_{1,i}]$ moves, for $i \in \{1,2\}$.

(1) Suppose $C[R_{1,1}]$ moves. Since n and m are ST, by Theorem 5.12 neither P_2 nor Q_2 can exercise any capability. Hence $R_{1,1} \not\xrightarrow{\alpha}$ for all $\alpha \neq \tau$. Since any observable $C[R_{1,1}] \downarrow_h$ originates from the context, we also have $C[R_{2,1}] \downarrow_h$.
We now look at the match on τ-transitions. There are two cases depending on whether $R_{1,1}$ contributes to the action. The more interesting case is when $R_{1,1}$ contributes to the action. (In the other case the proof is similar to the corresponding case in the proof of Lemma C.3.)
Suppose $R_{1,1}$ contributes to the action. By Theorem 5.12 no ambient can enter inside n and m, ambients n and m cannot be opened, no ambient of P_2 can exit from n, and n cannot move. Furthermore, Q_2 cannot exercise any capability. Condition $P_2 \not\stackrel{\mathtt{out}\,n}{\Longrightarrow}$ guarantees also that no ambient of P_2 can exit from n. We conclude that no interaction between $R_{1,1}$ and the context is possible.
Hence, $R' = C[R'']$ with $R_{1,1} \xrightarrow{\tau} R''$. This means that either $R'' = R_{2,1}$ or $R'' = n[\,\overline{\mathtt{out}}\,n.\,P_1 \mid m[\,\mathtt{out}\,n.\,Q_1 \mid Q_2'\,] \mid P_2\,]$ with $Q_2 \xrightarrow{\tau} Q_2'$, or $R'' = n[\,\overline{\mathtt{out}}\,n.\,P_1 \mid m[\,\mathtt{out}\,n.\,Q_1 \mid Q_2\,] \mid P_2'\,]$ with $P_2 \xrightarrow{\tau} P_2'$.
When $R'' = R_{2,1}$ we are done because of the identity relation $\mathcal{I}$. In the second case, we have $R_{2,1} \xrightarrow{\tau} R'''$, where $R''' = m[\,Q_1 \mid Q_2'\,] \mid n[\,P_1 \mid P_2\,]$, and also $C[R_{2,1}] \xrightarrow{\tau} C[R''']$. We conclude that $C[R'']\ \mathcal{R}_1\ C[R''']$, because by subject reduction both $\Gamma \vdash C[R''] : \mathit{Proc}$ and $\Gamma \vdash C[R'''] : \mathit{Proc}$ hold. The remaining case is similar, using also $P_2' \not\stackrel{\mathtt{out}\,n}{\Longrightarrow}$.

(2) Suppose $C[R_{1,2}]$ moves. Since $\Gamma(n)$ is of the form $\mathtt{STAmb}\,I\,[\uparrow\ W]$, then (by Theorem 5.12) process $\langle M\rangle \mid (x)P \mid Q$ has at most one unguarded capability or message; this implies that process Q has no unguarded capabilities and no unguarded messages (it also has no unguarded abstractions, by hypothesis). Hence $R_{1,2} \not\xrightarrow{\alpha}$ for all $\alpha \neq \tau$. Since $R_{1,2} \not\downarrow_h$ for every h, any observable $C[R_{1,2}] \downarrow_h$ originates from the context; therefore also $C[R_{2,2}] \downarrow_h$ holds.
We now look at the matching of τ-transitions. There are two cases depending on whether $R_{1,2}$ contributes to the action. When $R_{1,2}$ does not contribute to the action, the proof is similar to the corresponding case in the proof of Lemma C.3.
Suppose $R_{1,2}$ contributes to the action. By Theorem 5.12, no ambient can enter n and exit from n, ambient n cannot be opened, and n cannot move. Furthermore, message M cannot be consumed by Q, because it does not have unguarded abstractions. In conclusion, no interaction between $R_{1,2}$ and the context is possible.

Hence, $R' = C[R'']$ with $R_{1,2} \xrightarrow{\tau} R''$. This means that either $R'' = R_{2,2}$ or $R'' = n[\,\langle M\rangle \mid (x)P \mid Q'\,]$ with $Q \xrightarrow{\tau} Q'$.
The former case is immediate because of the identity relation $\mathcal{I}$. In the latter case, we have $R_{2,2} \xrightarrow{\tau} R'''$, where $R''' = n[\,P\{{}^{M}\!/\!{}_{x}\} \mid Q'\,]$, and also $C[R_{2,2}] \xrightarrow{\tau} C[R''']$. By subject reduction both $\Gamma \vdash C[R''] : \mathit{Proc}$ and $\Gamma \vdash C[R'''] : \mathit{Proc}$ hold. Moreover, by Theorem 5.12 and by Lemma C.4, Q' does not have unguarded abstractions. We conclude that $C[R'']\ \mathcal{R}_2\ C[R''']$.

□

Lemma 7.5 follows from Lemma C.8, Lemma C.10 and Lemma C.9.

LEMMA C.8. *Let $\mathcal{R} = \{(C[R_1], C[R_2])\}$ be the process relation such that:*

—*C is a context that respects the linearity of $L = \{\overline{\mathtt{in}}\,n\}$;*

—*$\Gamma \vdash C[R_1] : \mathit{Proc}$ where n is immobile and m is ST in Γ;*

—*$R_1 = m[\,\mathtt{in}\,n.\,P_1 \mid P_2\,] \mid n[\,!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2\,]$ and $R_2 = n[!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid Q_2 \mid m[P_1 \mid P_2]]$ where P_i, Q_i $(i = 1, 2)$ do not know $\overline{\mathtt{in}}\,n$.*

Relation $\mathcal{I} \cup \mathcal{R}$ is a weak barbed bisimulation with respect to L (up to $\equiv$).

PROOF. The proof is similar to that of case (1) in the proof of Lemma C.6 with a main difference: ambient n may interact with the context. In particular, ambients (of the context) may enter inside n and ambients may also go out of n, because n is immobile rather than $ST$and the context may use $\mathtt{in}\,n$.

Let $C[R_1]\ \mathcal{R}\ C[R_2]$. The most interesting part of the proof is to show how $C[R_2]$ matches the moves of $C[R_1]$.

First, we show how the observables of $C[R_1]$ are also observables of $C[R_2]$. Since m is ST, then (by Theorem 5.12) P_2 cannot exercise any capability. Hence, $m[\,\mathtt{in}\,n.\,P_1 \mid P_2\,] \xrightarrow{\alpha} O$ with $\alpha \neq \tau$ if and only if $\alpha = \mathtt{in}\,n$ and $O = \langle m[\,P_1 \mid P_2\,]\rangle\mathbf{0}$. Process Q_2 cannot exercise $\overline{\mathtt{in}}\,n$, because it does not know $\overline{\mathtt{in}}\,n$, and also cannot exercise any capability besides $\overline{\mathtt{out}}\,n$ and $\mathtt{open}\,k$ for some name k, because n is immobile. Thus, $n[\,!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2\,] \xrightarrow{\alpha} O$ with $\alpha \neq \tau$ iff $\alpha = \overline{\mathtt{in}}\,n$ and $O = \langle !\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid Q_2\rangle\mathbf{0}$. We therefore have $R_1 \xrightarrow{\alpha} O$ with $\alpha \neq \tau$ iff one of the following cases hold:

—$O \equiv \langle !\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid Q_2\rangle(m[\,\mathtt{in}\,n.\,P_1 \mid P_2\,])$ and $\alpha = \overline{\mathtt{in}}\,n$,

—$O \equiv \langle m[\,P_1 \mid P_2\,]\rangle(n[\,!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2\,])$ and $\alpha = \mathtt{in}\,n$.

We notice that we have also

$$R_2 \xrightarrow{\overline{\mathtt{in}}\,n} \equiv \langle !\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid Q_1 \mid Q_2 \mid m[\,P_1 \mid P_2\,]\rangle\mathbf{0}.$$

Therefore, both $R_1 \downarrow_n^L$ and $R_2 \downarrow_n^L$ hold, while $R_1 \not\downarrow_h^L$ for all $h \neq n$, so that any observable $C[R_1] \downarrow_h^L$ originates either from the context or, for $h = n$, from R_1. Since $R_2 \downarrow_n^L$, we conclude that if $C[R_1] \downarrow_h^L$ then also $C[R_2] \downarrow_h^L$, for all names h.

We now look at the matching of τ-transitions. We have two cases depending on whether R_1 contributes to the action. The most interesting case is when R_1 contributes to the action. The other case is similar to the corresponding case in the proof of Lemma C.3.

Suppose $C[R_1] \xrightarrow{\tau} R$ and R_1 contributes to the action. By Theorem 5.12 no ambient can enter m or exit from m, ambient m cannot be opened, and P_2 cannot exercise any capability. Furthermore, since the context does not know $\overline{\mathtt{in}}\,n$, ambient m cannot move inside any ambient n of the context. By Theorem 5.11, Q_2 cannot exercise any capability of movement and n cannot be opened. In conclusion, we have the following possibilities:

(1) The context C does not contribute to the action; that is, $R = C[R']$ with $R_1 \xrightarrow{\tau} R'$. We are either in a case that is similar to one of those analised in case (1) of Lemma C.6, or an ambient goes out of n, that is,

$$R' \equiv \boldsymbol{\nu}\tilde{p}\,(S \mid m[\,\mathtt{in}\,n.\,P_1 \mid P_2\,] \mid n[\,!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2'\,]).$$

In the latter case, transition $R_1 \xrightarrow{\tau} \equiv R'$ has been obtained by rule par using as premise

$$n[\,!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2\,] \xrightarrow{\tau} \boldsymbol{\nu}\tilde{p}\,(S \mid n[\,!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2'\,]).$$

By definition of the labeled transitions, transition $n[\,!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2\,] \xrightarrow{\tau} \boldsymbol{\nu}\tilde{p}\,(S \mid n[\,!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2'\,])$ has been obtained by applying rule tau-out to the premise

$$!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2 \xrightarrow{?n} \boldsymbol{\nu}\tilde{p}\,\langle S\rangle(!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2').$$

Moreover, transition $!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2 \xrightarrow{?n} \boldsymbol{\nu}\tilde{p}\langle S\rangle(!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2')$ has been obtained by applying rule par to the premise

$$Q_2 \xrightarrow{?n} \boldsymbol{\nu}\tilde{p}\,\langle S\rangle Q_2'.$$

Using the premise $Q_2 \xrightarrow{?n} \boldsymbol{\nu}\tilde{p}\,\langle S\rangle Q_2'$, by rule par we obtain a transition

$$!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid m[\,P_1 \mid P_2\,] \mid Q_2 \xrightarrow{?n} \boldsymbol{\nu}\tilde{p}\,\langle S\rangle(!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid m[\,P_1 \mid P_2\,] \mid Q_2')$$

and, by rule tau-out,

$$R_2 \xrightarrow{\tau} R''$$

where $R'' \equiv \boldsymbol{\nu}\tilde{p}\,(S \mid n[\,!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid m[\,P_1 \mid P_2\,] \mid Q_2'\,])$.
Let D be the context $\boldsymbol{\nu}\tilde{p}\,(S \mid [\cdot])$ and let $S_1 = n[\,!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_2'\,]$ and $S_2 = n[\,!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid m[\,P_1 \mid P_2\,] \mid Q_2'\,]$.
We have $C[R_2] \xrightarrow{\tau} \equiv C[D[S_2]]$ and also $C[R_1] \xrightarrow{\tau} \equiv C[D[S_1]]$. By subject reduction both $\Gamma \vdash C[D[S_1]] : \mathit{Proc}$ and $\Gamma \vdash C[D[S_2]] : \mathit{Proc}$ hold. Since D does not know $\overline{\mathtt{in}}\,n$, we conclude that $C[D[S_1]]\ \mathcal{R}\ C[D[S_2]]$ (up to $\equiv$).

(2) Both R_1 and the context contributes to the action. Arguing by induction on the depth of the inference, the last rule used to obtain transition $C[R_1] \xrightarrow{\tau} R$ may have been tau-in, tau-amb, res, par. Note that rule tau-open could not have been applied, since ambients n and m cannot be opened. Moreover, rule tau-out could not have been applied, since both n and m cannot go out of any ambient.
We show only the case of tau-in; the remaining cases are immediate by induction hypothesis. We now notice that ambient n does not move, ambient m cannot move inside any ambient of the context and $\overline{\mathtt{in}}\,n$ is uniform receptive.

Therefore, an ambient of the context enters inside n (by consuming a coaction $\overline{\mathtt{in}}\,n$ not appearing in Q_2).
By definition of labeled transitions we have $C[R_1] = P \mid C_1[R_1]$ and $R \equiv C_2[R']$ with

$$P \xrightarrow{\mathtt{in}\,n} \boldsymbol{\nu}\tilde{p}_1\,\langle S\rangle S_1$$

$$C_1[R_1] \xrightarrow{\overline{\mathtt{in}}\,n} \boldsymbol{\nu}\tilde{p}_2\,\langle(!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid Q_2)\rangle(m[\,\mathtt{in}\,n.\,P_1 \mid P_2\,] \mid S_2)$$

where $C_2 = \boldsymbol{\nu}\tilde{p}_1, \tilde{p}_2\,([\cdot] \mid S_1 \mid S_2)$, and $R' = m[\,\mathtt{in}\,n.\,P_1 \mid P_2\,] \mid n[\,!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid Q_2 \mid S\,]$.
Moreover, since $\overline{\mathtt{in}}\,n$ is uniformly receptive, we deduce that transition

$$C_1[R_1] \xrightarrow{\overline{\mathtt{in}}\,n} \boldsymbol{\nu}\tilde{p}_2\,\langle(!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid Q_2)\rangle(m[\,\mathtt{in}\,n.\,P_1 \mid P_2\,] \mid S_2)$$

has been obtained by applying the rules of the labeled semantics from the premise

$$R_1 \xrightarrow{\overline{\mathtt{in}}\,n} \langle(!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid Q_2)\rangle(m[\,\mathtt{in}\,n.\,P_1 \mid P_2\,]).$$

Since also

$$R_2 \xrightarrow{\overline{\mathtt{in}}\,n} \langle(!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid Q_1 \mid m[\,P_1 \mid P_2\,] \mid Q_2)\rangle\mathbf{0}$$

we derive

$$C_1[R_2] \xrightarrow{\overline{\mathtt{in}}\,n} \boldsymbol{\nu}\tilde{p}_2\,\langle(!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid Q_1 \mid Q_2 \mid m[\,P_1 \mid P_2\,])\rangle(S_2).$$

By rule tau-in, using as premises

$$P \xrightarrow{\mathtt{in}\,n} \boldsymbol{\nu}\tilde{p}_1\,\langle S\rangle S_1$$

$$C_1[R_2] \xrightarrow{\overline{\mathtt{in}}\,n} \boldsymbol{\nu}\tilde{p}_2\,\langle(!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid Q_1 \mid Q_2 \mid m[\,P_1 \mid P_2\,])\rangle(S_2)$$

we therefore obtain

$$C[R_2] \xrightarrow{\tau} \equiv C_2[R'']$$

where $R'' = n[\,!\overline{\mathtt{in}}\,n.\,Q_1 \mid Q_1 \mid Q_1 \mid Q_2 \mid m[\,P_1 \mid P_2\,] \mid S\,]$.
Hence, we have $C[R_1] \xrightarrow{\tau}\equiv C_2[R']$ and $C[R_2] \xrightarrow{\tau}\equiv C_2[R'']$. By subject reduction both $\Gamma \vdash C_2[R'] : \mathit{Proc}$, and $\Gamma \vdash C_2[R''] : \mathit{Proc}$ hold. Since C_2 and S do not know $\overline{\mathtt{in}}\,n$, we conclude that $C_2[R']\ \mathcal{R}\ C_2[R'']$ (up to $\equiv$).

□

LEMMA C.9. *Let $\mathcal{R} = \{(C[R_1], C[R_2])\}$ be the process relation such that:*

—*C is a context that respects the linearity of $\{\mathtt{open}\,m\}$;*

—*$\Gamma \vdash C[R_1] : \mathit{Proc}$ where n is immobile and m is ST in Γ;*

—*$R_1 = n[\,!\mathtt{open}\,m.\,P_1 \mid m[\,\overline{\mathtt{open}}\,m.\,Q_1 \mid Q_2\,] \mid P_2\,]$ and $R_2 = n[!\mathtt{open}\,m.\,P_1 \mid P_1 \mid Q_1 \mid Q_2 \mid P_2]$, where P_i, Q_i $(i = 1, 2)$ do not know $\mathtt{open}\,m$.*

Relation $\mathcal{I} \cup R$ is a weak barbed bisimulation with respect to $\{\mathtt{open}\,m\}$ (up to $\equiv$).

PROOF. The proof is similar to those of case (2) in the proof of Lemma C.6 and case (4) in the proof of Lemma C.3 with a main difference: ambient n may interact with the context and the capability $\mathsf{open}\,m$ may be consumed by other ambients m placed inside n. In particular, ambients (of the context) may enter inside n, and ambients may go out of n, because n is immobile.

Let $C[R_1]\ \mathcal{R}\ C[R_2]$. The most interesting part of the proof is to show how $C[R_2]$ matches the moves from $C[R_1]$.

First, we show how the observables of $C[R_1]$ are also observables of $C[R_2]$. Since m is ST, then (by Theorem 5.12) Q_2 cannot exercise any capability. Hence, $m[\,\overline{\mathsf{open}}\,m.\,Q_1 \mid Q_2\,] \xrightarrow{\alpha} O$ with $\alpha \neq \tau$ if and only if $\alpha = ?\mathsf{open}\,m$ and $O = Q_1 \mid Q_2$. Since n is immobile and P_2 does not know $\mathsf{open}\,m$, process P_2 cannot exercise any capability besides $\overline{\mathtt{in}}\,n$, $\overline{\mathtt{out}}\,n$, and $\mathsf{open}\,k$ for names $k \neq m$. Therefore, we have $R_1 \xrightarrow{\alpha} O$ with $\alpha \neq \tau$ if and only if $\alpha = \overline{\mathtt{in}}\,n$ and $O = \langle !\mathsf{open}\,m.\,P_1 \mid m[\,\overline{\mathsf{open}}\,m.\,Q_1 \mid Q_2\,] \mid P_2'\rangle\mathbf{0}$. By definition of the labeled transition semantics, transition

$$R_1 \xrightarrow{\overline{\mathtt{in}}\,n} \langle !\mathsf{open}\,m.\,P_1 \mid m[\,\overline{\mathsf{open}}\,m.\,Q_1 \mid Q_2\,] \mid P_2'\rangle\mathbf{0}$$

has been obtained, by rule co-in, using as premise

$$!\mathsf{open}\,m.\,P_1 \mid m[\,\overline{\mathsf{open}}\,m.\,Q_1 \mid Q_2\,] \mid P_2 \xrightarrow{\overline{\mathtt{in}}\,n} !\mathsf{open}\,m.\,P_1 \mid m[\,\overline{\mathsf{open}}\,m.\,Q_1 \mid Q_2\,] \mid P_2'.$$

Moreover, transition

$$!\mathsf{open}\,m.\,P_1 \mid m[\,\overline{\mathsf{open}}\,m.\,Q_1 \mid Q_2\,] \mid P_2 \xrightarrow{\overline{\mathtt{in}}\,n} !\mathsf{open}\,m.\,P_1 \mid m[\,\overline{\mathsf{open}}\,m.\,Q_1 \mid Q_2\,] \mid P_2'$$

has been obtained, by rule par, using as premise $P_2 \xrightarrow{\overline{\mathtt{in}}\,n} P_2'$. We now notice that, by rule par, using as premise $P_2 \xrightarrow{\overline{\mathtt{in}}\,n} P_2'$, we also obtain

$$!\mathsf{open}\,m.\,P_1 \mid P_1 \mid Q_1 \mid Q_2 \mid P_2 \xrightarrow{\overline{\mathtt{in}}\,n} !\mathsf{open}\,m.\,P_1 \mid P_1 \mid Q_1 \mid Q_2 \mid P_2'.$$

and therefore, by applying rule co-in, we derive

$$R_2 \xrightarrow{\overline{\mathtt{in}}\,n} \langle !\mathsf{open}\,m.\,P_1 \mid P_1 \mid Q_1 \mid Q_2 \mid P_2'\rangle\mathbf{0}.$$

Since $\overline{\mathtt{in}}\,n$ is observable, if $P_2 \xrightarrow{\overline{\mathtt{in}}\,n} P_2'$ then both $R_1 \downarrow^L_n$ and $R_2 \downarrow^L_n$ hold. Since $R_1 \not\downarrow^L_h$ for any $h \neq n$, any other possible observable of $C[R_1]$ originates from the context. Therefore, we can conclude that if $C[R_1] \downarrow^L_h$, for some h, then also $C[R_2] \downarrow^L_h$.

We now consider τ-transitions from $C[R_1]$, and how $C[R_2]$ can match it. We have two cases depending on whether R_1 contributes to the action. The most interesting case is when R_1 contributes to the action. The proof of other case is similar to the corresponding case in the proof of Lemma C.3.

Let $C[R_1] \xrightarrow{\tau} R$ and suppose R_1 contributes to the action. Since n is immobile, by Theorem 5.11, ambients can exit and enter n, while n cannot move and cannot be opened. On the other hand, since m is ST, and both the context and P_2 do not know $\mathsf{open}\,m$, ambient m cannot be involved in other interactions. In conclusion, we have the following possibilities:

(1) The context C does not contribute to the action. Therefore, $R = C[R']$ with $R_1 \xrightarrow{\tau} R'$. We are then either in a case similar to one of those analised in case

(2) of Lemma C.6, or an ambient has gone out of n, or an ambient m of P_2 has been opened. Notice that m cannot be opened by processes of P_2, since $\mathsf{open}\, m$ is uniformly receptive. The former case is similar to the corresponding one analised in the proof of Lemma C.8.
When an ambient m of P_2 is opened, we have

$$R' \equiv n[\,!\mathsf{open}\, m.\, P_1 \mid P_1 \mid m[\,\overline{\mathsf{open}}\, m.\, Q_1 \mid Q_2\,] \mid P_2'\,].$$

By definition of the labeled transition semantics, transition $R_1 \xrightarrow{\tau} R'$ has been obtained, by rule tau-amb, using as premise

$$\begin{array}{l} !\mathsf{open}\, m.\, P_1 \mid m[\,\overline{\mathsf{open}}\, m.\, Q_1 \mid Q_2\,] \mid P_2 \xrightarrow{\tau} \\ \quad !\mathsf{open}\, m.\, P_1 \mid P_1 \mid m[\,\overline{\mathsf{open}}\, m.\, Q_1 \mid Q_2\,] \mid P_2'. \end{array}$$

Moreover, transition

$$\begin{array}{l} !\mathsf{open}\, m.\, P_1 \mid m[\,\overline{\mathsf{open}}\, m.\, Q_1 \mid Q_2\,] \mid P_2 \xrightarrow{\tau} \\ \quad !\mathsf{open}\, m.\, P_1 \mid P_1 \mid m[\,\overline{\mathsf{open}}\, m.\, Q_1 \mid Q_2\,] \mid P_2' \end{array}$$

has been obtained, by rule tau-open, using as premises

$$P_2 \xrightarrow{?\mathsf{open}\, m} P_2'$$

$$!\mathsf{open}\, m.\, P_1 \mid m[\,\overline{\mathsf{open}}\, m.\, Q_1 \mid Q_2\,] \xrightarrow{\mathsf{open}\, m} !\mathsf{open}\, m.\, P_1 \mid P_1 \mid m[\,\overline{\mathsf{open}}\, m.\, Q_1 \mid Q_2\,].$$

We have also

$$!\mathsf{open}\, m.\, P_1 \mid P_1 \mid Q_1 \mid Q_2 \xrightarrow{\mathsf{open}\, m} !\mathsf{open}\, m.\, P_1 \mid P_1 \mid P_1 \mid Q_1 \mid Q_2$$

and therefore, by applying rule tau-open, and rule tau-amb as above, we obtain

$$R_2 \xrightarrow{\tau} n[\,!\mathsf{open}\, m.\, P_1 \mid P_1 \mid P_1 \mid Q_1 \mid Q_2 \mid P_2'\,].$$

Hence, we have $C[R_1] \xrightarrow{\tau}\equiv C[R']$ and also $C[R_2] \xrightarrow{\tau}\equiv C[R'']$ with

$$R'' = n[\,!\mathsf{open}\, m.\, P_1 \mid P_1 \mid P_1 \mid Q_1 \mid Q_2 \mid P_2'\,]\,.$$

By subject reduction both $\Gamma \vdash C[R'] : \mathit{Proc}$, and $\Gamma \vdash C[R''] : \mathit{Proc}$ hold. We conclude that $C[R']\ \mathcal{R}\ C[R'']$ (up to $\equiv$).

(2) Both R_1 and the context contributes to the action. Arguing by induction on the depth of the inference, the last rule used to obtain transition $C[R_1] \xrightarrow{\tau} R$ may have been tau-in, res, par. Note that rule tau-open could not have been applied, since n cannot be opened. Moreover, rule tau-out could not have been applied, since n cannot move.
We show only the case of tau-in; the other cases are immediate by induction hypothesis. Since n is immobile, the only possibility is that an ambient of the context enters inside n.
This means that $C[R_1] = P \mid C_1[R_1]$ and $R \equiv C_2[R']$ with

$$P \xrightarrow{\mathsf{in}\, n} \boldsymbol{\nu}\tilde{p}_1\, \langle S \rangle S_1$$

$$C_1[R_1] \xrightarrow{\overline{\mathsf{in}}\, n} \boldsymbol{\nu}\tilde{p}_2\, \langle !\mathsf{open}\, m.\, P_1 \mid m[\,\overline{\mathsf{open}}\, m.\, Q_1 \mid Q_2\,] \mid P_2' \rangle S_2,$$

where $C_2 = \boldsymbol{\nu}\tilde{p}_1, \tilde{p}_2\ ([\cdot] \mid S_1 \mid S_2)$ and $R' = n[\,!\mathsf{open}\,m.\,P_1 \mid m[\,\overline{\mathsf{open}}\,m.\,Q_1 \mid Q_2\,] \mid P_2' \mid S\,]$.

By definition of the labeled transitions, transition

$$C_1[R_1] \xrightarrow{\overline{\mathsf{in}}\,n} \boldsymbol{\nu}\tilde{p}_2\ \langle !\mathsf{open}\,m.\,P_1 \mid m[\,\overline{\mathsf{open}}\,m.\,Q_1 \mid Q_2\,] \mid P_2'\rangle S_2$$

has been obtained, by applying the rules of the labeled transitions, from the premise

$$R_1 \xrightarrow{\overline{\mathsf{in}}\,n} \langle !\mathsf{open}\,m.\,P_1 \mid m[\,\overline{\mathsf{open}}\,m.\,Q_1 \mid Q_2\,] \mid P_2'\rangle \mathbf{0}\,.$$

Moreover, transition $R_1 \xrightarrow{\overline{\mathsf{in}}\,n} \langle !\mathsf{open}\,m.\,P_1 \mid m[\,\overline{\mathsf{open}}\,m.\,Q_1 \mid Q_2\,] \mid P_2'\rangle \mathbf{0}$ has been obtained, by applying rules co-in and par, from the premise $P_2 \xrightarrow{\overline{\mathsf{in}}\,n} P_2'$.

Since $P_2 \xrightarrow{\overline{\mathsf{in}}\,n} P_2'$, by rule par and co-in as above, we derive

$$R_2 \xrightarrow{\overline{\mathsf{in}}\,n} \langle !\mathsf{open}\,m.\,P_1 \mid P_1 \mid Q_1 \mid Q_2 \mid P_2'\rangle \mathbf{0}$$

and therefore also

$$C_1[R_2] \xrightarrow{\overline{\mathsf{in}}\,n} \boldsymbol{\nu}\tilde{p}_2\ \langle !\mathsf{open}\,m.\,P_1 \mid P_1 \mid Q_1 \mid Q_2 \mid P_2'\rangle S_2.$$

Finally, we obtain $C[R_2] \xrightarrow{\tau} C_2[R'']$, where $R'' = n[\,!\mathsf{open}\,m.\,P_1 \mid P_1 \mid Q_1 \mid Q_2 \mid P_2' \mid S\,]$, by rule tau-com using as premises

$$P \xrightarrow{\mathsf{in}\,n} \boldsymbol{\nu}\tilde{p}_1\ \langle S\rangle S_1$$

$$C_1[R_2] \xrightarrow{\overline{\mathsf{in}}\,n} \boldsymbol{\nu}\tilde{p}_2\ \langle !\mathsf{open}\,m.\,P_1 \mid P_1 \mid Q_1 \mid Q_2 \mid P_2'\rangle S_2.$$

Hence, we have $C[R_1] \xrightarrow{\tau} \equiv C_2[R']$ and also $C[R_2] \xrightarrow{\tau} \equiv C_2[R'']$. By subject reduction, both $\Gamma \vdash C_2[R'] : \mathit{Proc}$, and $\Gamma \vdash C_2[R''] : \mathit{Proc}$ hold. Since C_2 and S do not know $\mathsf{open}\,m$, we conclude that $C_2[R']\ \mathcal{R}\ C_2[R'']$ (up to $\equiv$).

□

Lemma C.10. *Let $\mathcal{R} = \{(C[R_1], C[R_2])\}$ be the process relation such that:*

—*C is a context that respects the linearity of $L = \{\overline{\mathsf{out}}\,n\}$;*

—*$\Gamma \vdash C[R_1] : \mathit{Proc}$ where n is immobile and m is ST in Γ;*

—*$R_1 = n[\,!\overline{\mathsf{out}}\,n.\,P_1 \mid m[\,\mathsf{out}\,n.\,Q_1 \mid Q_2\,] \mid P_2\,]$ and $R_2 = m[Q_1 \mid Q_2] \mid n[!\overline{\mathsf{out}}\,n.\,P_1 \mid P_1 \mid P_2]$ where P_i, Q_i with $i = 1, 2$* do not know *$\overline{\mathsf{out}}\,n$.*

Relation $\mathcal{I} \cup R$ is a weak barbed bisimulation *with respect to Γ and L (up to $\equiv$).*

Proof. The proof is similar to that of case (1) of Lemma C.7. Intuitively, the crux of the proof is: since m is ST, no ambient can enter and exit from m, ambient m cannot be opened, and Q_2 cannot exercise any capability; since n is immobile and P_2 does not know $\overline{\mathsf{out}}\,n$, process P_2 can exercise only capabilities $\overline{\mathsf{in}}\,n$ and $\mathsf{open}\,k$, for names $k \neq n$. Hence, the only interactions between R_1 and the context are when an ambient of the context enters inside n, or an ambient of P_2 exits from n. Therefore, it is enough to extend the proof of case (1) of Lemma C.7 with the two cases mentioned above, which are indeed similar to the corresponding cases in the proofs of Lemma C.8 and C.9.

□

REFERENCES

AMADIO, R. M. AND CARDELLI, L. 1993. Subtyping recursive types. *ACM Trans. Program. Lang. Syst. 15,* 4, 575–631. A preliminary version appeared in POPL '91 (pp. 101–118), and as DEC Systems Reasearch Center Reasearch Report number 62, August 1990.

AMTOFT, T., KFOURY, A. J., AND PERICAS-GEERTSEN, S. M. 2001. What are Polymorphically-Typed Ambients? In *Proc. of ESOP '01*. Lecture Notes in Computer Science, vol. 2028. Springer Verlag, Berlin, 206–220.

BUGLIESI, M. AND CASTAGNA, G. 2001. Secure Safe Ambients. In *Proceedings of the 28th Annual Symposium on Principles of Programming Languages*. ACM, New York.

CARDELLI, L. 1996. *Type Systems, Handbook of Computer Science and Engineering*. CRC Press.

CARDELLI, L., GHELLI, G., AND GORDON, A. D. 1999. Mobility types for mobile ambients. In *Proc. of ICALP '99*. Lecture Notes in Computer Science, vol. 1644. Springer Verlag, Berlin, 230–239.

CARDELLI, L. AND GORDON, A. D. 1998a. Mobile ambients. In *Proc. of FoSSaCS '98*, M. Nivat, Ed. Lecture Notes in Computer Science, vol. 1378. Springer Verlag, Berlin, 140–155.

CARDELLI, L. AND GORDON, A. D. 1998b. Technical annex to [Cardelli and Gordon 1998a]. Unpublished notes.

CARDELLI, L. AND GORDON, A. D. 1999a. Equational properties of mobile ambients. In *Proc. of FoSSaCS '99*, W. Thomas, Ed. Lecture Notes in Computer Science, vol. 1578. Springer Verlag, Berlin, 212–226.

CARDELLI, L. AND GORDON, A. D. 1999b. Equational properties of mobile ambients. Technical Report MSR-TR-99-11, Microsoft Research.

CARDELLI, L. AND GORDON, A. D. 1999c. Types for mobile ambients. In *Proceedings of the 26th Annual Symposium on Principles of Programming Languages*. ACM, New York, 79–92.

DEGANO, P., LEVI, F., AND BODEI, C. 2000. Safe Ambients: Control Flow Analysis and Security. In *Proc. of ASIAN '00*. Lecture Notes in Computer Science, vol. 1961. Springer Verlag, Berlin, 199–214.

DEZANI-CIANCAGLINI, M. AND SALVO, I. 2000. Security types for mobile safe ambients. In *Proc. of ASIAN '00*. Lecture Notes in Computer Science, vol. 1961. Springer Verlag, Berlin, 215–236.

FOURNET, C., GONTHIER, G., LÉVY, J.-J., MARANGET, L., AND RÉMY, D. 1996. A calculus of mobile processes. In *Proc. of CONCUR '96*, U. Montanari and V. Sassone, Eds. Lecture Notes in Computer Science, vol. 1119. Springer Verlag, Berlin, 406–421.

GADDUCCI, F. AND MONTANARI, U. 2001. A concurrent graph semantics for mobile ambients. Electronic Notes in Computer Science, vol. 45. Elsevier.

GUAN, X., YANG, Y., AND YOU, J. 2000. Making ambients more robust. In *Proc. Intl. Conf. on Software: Theory and Practice*. 377–384.

GUAN, X., YANG, Y., AND YOU, J. 2001. Typing evolving ambients. *Information Processing Letters 80,* 5, 265–270.

HENNESSY, M. AND RIELY, J. 1998. A typed language for distributed mobile processes. In *Proceedings of the 25th Annual Symposium on Principles of Programming Languages*. ACM, New York, 378–390.

KOBAYASHI, N., PIERCE, B., AND TURNER, D. 1996. Linearity and the pi-calculus. In *Proceedings of the 23th Annual Symposium on Principles of Programming Languages*. ACM, New York, 358–371.

MERRO, M. AND HENNESSY, M. 2002. Bisimulation congruences in Safe Ambients. In *Proceedings of the 29th Annual Symposium on Principles of Programming Languages*. ACM, New York.

MILNER, R. 1989. *Communication and Concurrency*. Prentice Hall.

MILNER, R., PARROW, J., AND WALKER, D. 1992. A calculus of mobile processes, (Parts I and II). *Information and Computation 100*, 1–77.

MILNER, R. AND SANGIORGI, D. 1992. Barbed bisimulation. In *Proc. of ICALP '92*, W. Kuich, Ed. Lecture Notes in Computer Science, vol. 623. Springer Verlag, Berlin, 685–695.

NIELSON, H. R., NIELSON, F., HANSEN, R. R., , AND JENSEN, J. G. 1999. Validating firewalls in mobile ambients. In *Proc. of CONCUR '99*, J. Baeten and S. Mauw, Eds. Lecture Notes in Computer Science, vol. 1664. Springer Verlag, Berlin, 463–478.

PIERCE, B. AND SANGIORGI, D. 1996. Typing and subtyping for mobile processes. *Journal of Mathematical Structures in Computer Science 6,* 5, 409–454. An extended abstract appeared in *Proc. LICS 93*, IEEE Computer Society Press.

REYNOLDS, J. 1978. Syntactic control of interference. In *Proceedings of the 5th Annual Symposium on Principles of Programming Languages*. ACM, New York, 39–46.

SANGIORGI, D. 1997. The name discipline of uniform receptiveness. In *Proc. of ICALP '97*. Lecture Notes in Computer Science, vol. 1256. Springer Verlag, Berlin.

SANGIORGI, D. AND VALENTE, A. 2001. A distributed abstract machine for Safe Ambients. In *Proc. of ICALP '01*. Lecture Notes in Computer Science, vol. 2076. Springer Verlag, Berlin.

SANGIORGI, D. AND WALKER, D. 2001. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press.

SEWELL, P. 1998. Global/local subtyping and capability inference for a distributed π-calculus. In *Proc. of ICALP '98*. Lecture Notes in Computer Science, vol. 1443. Springer Verlag, Berlin.

VITEK, J. AND CASTAGNA, G. 1999. Seal: A framework for secure mobile computations. In *Proc. Internet Programming Languages*. Lecture Notes in Computer Science, vol. 1686. Springer Verlag, Berlin.

ZIMMER, O. 2000. On the expressiveness of pure mobile ambients. In *Proc. of EXPRESS '00*, L. Aceto and B. Victor, Eds. Electronic Notes in Computer Science, vol. 39. Elsevier.