Math. Struct. in Comp. Science (2007), *vol.* 17, *pp.* 587–645. © 2007 Cambridge University Press
doi:10.1017/S0960129507006226 Printed in the United Kingdom

Boxed ambients with communication interfaces[†]

PABLO GARRALDA[‡], EDUARDO BONELLI[§],
ADRIANA COMPAGNONI[‡] and
MARIANGIOLA DEZANI-CIANCAGLINI[¶]

[‡]*Stevens Institute of Technology, Computer Science Department, Castle Point on Hudson, Hoboken, NJ 07030, U.S.A.*
Email: abc@cs.stevens.edu

[§]*Universidad Nacional de La Plata and CONICET, Facultad de Informática, LIFIA, Calle 115 entre 49 y 50, 1900 La Plata, Buenos Aires, Argentina*

[¶]*Università di Torino, Dipartimento di Informatica, Corso Svizzera, 185, 10149 Torino, Torino, Italy*

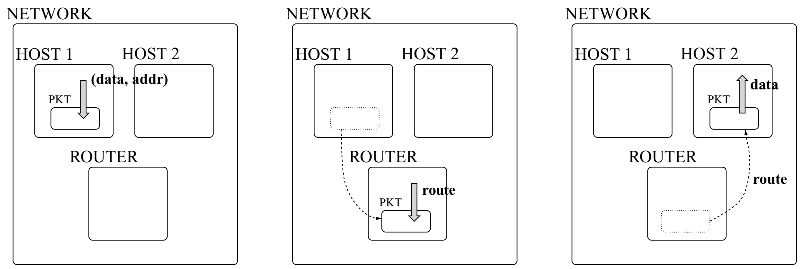
Received 14 November 2005; revised 1 October 2006

We define **BACI** *(Boxed Ambients with Communication Interfaces)*, an ambient calculus with a flexible communication policy. Traditionally, typed ambient calculi have a fixed communication policy determining the kind of information that can be exchanged with a parent ambient, even though mobility changes the parent. **BACI** lifts that restriction, allowing different communication policies with different parents during computation. Furthermore, **BACI** separates communication and mobility by making the channels of communication between ambients explicit. In contrast with other typed ambient calculi where communication policies are global, each ambient in **BACI** is equipped with a description of the communication policies ruling its information exchange with parent and child ambients. The communication policies of ambients increase when they move: more precisely, when an ambient enters another ambient, the entering ambient and the host ambient can exchange their communication ports and agree on the kind of information to be exchanged. This information is recorded locally in both ambients.
We show the type-soundness of **BACI**, proving that it satisfies the subject reduction property, and we study its behavioural semantics by means of a labelled transition system.

1. Introduction

In an ambient calculus, one can distinguish between two forms of dynamic behaviour: *communication* and *migration* (Cardelli and Gordon 2000). By communication, we mean the exchange of information between processes, which may be located in different ambients. By migration, we mean the ability of an ambient to relocate itself by entering or exiting other ambients. Communication and migration are deeply related since migration and communication may enable or disable each other.

[†] This research was partially supported by the EU FP6-2004-510996 Coordination Action TYPES, by MURST Cofin'04 project McTafi, and by the USA under the National Science Foundation project No. CCR-0220286 ITR: Secure Electronic Transactions. The funding bodies are not responsible for any use that might be made of the results presented here.



(a) Packet creation. (b) Packet routing. (c) Packet delivery.

Fig. 1. Example of an ambient using different TOCs with different parents.

In calculi such as BA (Bugliesi *et al.* 2004) and NBA (Bugliesi *et al.* 2005), and those in Castagna *et al.* (2005) and Merro and Sassone (2002), an ambient can communicate with its parent ambient (the host ambient) or with a child ambient (an ambient it contains), and there may also be local communication between the processes within an ambient. In typed ambient calculi, communication is controlled by types, and the type of information being exchanged is often called the *topic of conversation* (TOC). For example, if an ambient sends the number 3 to its parent, we can say that the TOC is Int.

Migration, entering or exiting an ambient, changes the parent of an ambient. Existing typed mobile ambient calculi fix a TOC for communication with the parent for each ambient, and the TOC remains fixed even if migration changes the parent.

For example, consider Figure 1, where HOST 1 needs to send **data** to HOST 2. HOST 1 does not know where HOST 2 is located, but it knows the location (**addr**) of a ROUTER that can forward to HOST 2 the packet (PKT) containing the **data**. Assuming this, HOST 1 spawns the packet and forwards the data to be transported along with the location of the router (Fig. 1(a)). Next, the packet moves inside the ROUTER, where it obtains the **route** to HOST 2 (Fig. 1(b)). Finally, using that **route**, the packet reaches HOST 2 and delivers **data** (Fig. 1(c)).

Note that the PKT ambient uses three different TOCs with its three different parents (that is, HOST 1, ROUTER and HOST 2). In order to implement this example in calculi where each ambient has a fixed type for parent communication, additional messenger ambients are needed to encode the communication with the different parents, using an auxiliary messenger ambient for each communication type. The use of these messenger ambients may lead to an overpopulation of ambients that makes the design of systems both error-prone and more difficult to understand.

The type systems of almost all ambient calculi assume that the communication behaviour of any ambient is known globally. This means that every ambient knows exactly what the communication types (that is, TOCs) of the rest of the ambients are. This assumption greatly simplifies reasoning with ambients; however, for distributed systems, this kind of global knowledge is not realistic. A more faithful model would allow each ambient to carry its own *local view* of its surroundings. Furthermore, in a *mobile* setting, such local views would be updateable, given that processes expand their local views as they move about.

In this paper we introduce **BACI**, a new mobile ambient calculus where each ambient carries a *communication interface* specifying how an ambient may interact with the environment. The design of **BACI** was driven by the desire to lift the restriction of a fixed TOC with parents, by allowing an ambient to change the TOC when changing parents and enabling the straightforward design of ambients that need to exchange information of different types with different ambients.

Moreover, **BACI** tackles the problem of describing the behaviour of systems that have local information regarding the usage of the communication channels. In **BACI**, each ambient is provided with its own *local view* of the communication behaviour of the rest of the system. This design provides both more flexibility and a framework that is closer to the implementation of distributed systems.

1.1. *Ports and names*

Communication with a child ambient is often labelled with the ambient's name (named communication). For example:

$$n\textbf{[}\ \langle 3\rangle^{\downarrow m} \mid m\textbf{[}\cdots\textbf{]} \mid \cdots\textbf{]} \quad \text{(ambient } n \text{ wants to send 3 to its child } m).$$

However, in communication with a parent, the name is often left implicit, since the parent can be uniquely determined by the location of an ambient.

$$n\textbf{[}\ m\textbf{[}\ \langle 3\rangle^{\uparrow} \mid \cdots\textbf{]} \mid \cdots\textbf{]} \quad \text{(ambient } m \text{ wants to send 3 to its parent } n).$$

In order to allow different TOCs with different parents along with local typing information, **BACI** introduces named communication with parents and the use of communication ports.

A natural choice, which is used in most ambient calculi, is to use ambient names to identify communication with ambients. However, in a setting with local typing information, this can be problematic: an ambient name that is received as a message might be used to reference the ambient's communication in a way that contradicts the local typing information. In the example

$$\begin{gathered}\langle m\rangle^{\downarrow n} \mid \ n\textbf{[}(x)^{\uparrow}.\langle 3\rangle^{\downarrow x} \mid \langle p\rangle^{\downarrow m} \mid m\textbf{[}\cdots\textbf{]}\,\textbf{]}\\ \longrightarrow \\ n\textbf{[}\langle 3\rangle^{\downarrow m} \mid \langle p\rangle^{\downarrow m} \mid m\textbf{[}\cdots\textbf{]}\,\textbf{]},\end{gathered}$$

when the message containing the name m is transmitted down to the ambient n, the variable x is replaced by m in the expression $\langle 3\rangle^{\downarrow x}$. If the local information in ambient n dictates that the communication type with child ambient m is not Int (that is, m expects ambient names from its parent), this substitution produces an inconsistency with the typing assumptions of n and a potential type mismatch in the communication between n and m.

Therefore, in order to gain control over communication, we introduce the concept of a communication port: each ambient is locally associated (in addition to its ambient name) with a communication port. In this way, the communication ports are used to exchange information, and the ambient names are kept purely for migration.

In this framework, an ambient n with communication port c_n is written $n[c_n \| \cdots]$. For example:

$$n[c_n \| \ \cdots \ | \ m[c_m \| \ \langle 3 \rangle^{\uparrow c_n}] \ | \ \cdots] \quad \text{(ambient } m \text{ wants to send 3 to parent port } c_n\text{).}$$

Communication and mobility are decoupled: an ambient's name denotes a location and an ambient's port denotes its unique communication channel. Notice that any combination of ports and ambient names is possible. This allows us to have non-determinism independently in either communication or mobility if, for instance, two ambients have the same port or the same name, respectively. Two ambients denoting different locations (that is, with different names) are indistinguishable from the communications point of view if they have the same port name.

The introduction of communication ports naturally leads us to associate TOCs with ports rather than, as usual, with ambient names. As we suggested earlier, there is no global knowledge of this association: each ambient has its *local view*, which associates communication ports with its communication behaviour. Moreover, *local views* can increase dynamically with relocation.

An ambient n with port c_n and local view Γ is written $n[\Gamma \| c_n \| \cdots]$. So, our last example becomes

$$n[\{\cdots, \mathsf{Int}^{\downarrow c_m}\} \| c_n \| \cdots \ | \ m[\{\cdots, \mathsf{Int}^{\uparrow c_n}\} \| c_m \| \langle 3 \rangle^{\uparrow c_n} \ | \ \cdots]].$$

In summary, each ambient in **BACI** comes equipped with its own *local communication interface*. A communication interface consists of:

— a *communication port* to exchange information with other ambients; and
— a *local view* associating topics of conversation to parent and children ports.

These two new ingredients allow

— different TOCs with different parents, and
— different TOCs with different children,

while sharing the same ambient name. Neither of these features is supported in any other BA calculus. Moreover, we will see in Subsection 1.3 that, when an ambient enters another ambient, they can exchange their port names, enriching in this way both their local views. For example, in the expression

$$m[\{\mathsf{Int}^{\downarrow c_n}\} \| c_m \| (x : \mathsf{Int})^{\downarrow c_n}.P \ | \ n[\{\mathsf{Int}^{\uparrow c_m}\} \| c_n \| \langle 3 \rangle^{\uparrow c_m}]],$$

the ambient m has local view $\{\mathsf{Int}^{\downarrow c_n}\}$ and communication port c_m, while the ambient n has local view $\{\mathsf{Int}^{\uparrow c_m}\}$ and communication port c_n. Here, n sends the integer 3 to its parent port c_m. Similarly, m reads a message x from the port c_n of its child ambient n. The local views of each participating ambient guarantee that the type of the message expected by m and the type of the value sent by n are compatible. In this example, since the local views are compatible (they both want to communicate information of type Int, and the channel names match), communication can take place without a risk of a type mismatch:

$$\begin{array}{c} m[\{\mathsf{Int}^{\downarrow c_n}\} \| c_m \| (x : \mathsf{Int})^{\downarrow c_n}.P \ | \ n[\{\mathsf{Int}^{\uparrow c_m}\} \| c_n \| \langle 3 \rangle^{\uparrow c_m}]] \\ \longrightarrow \\ m[\{\mathsf{Int}^{\downarrow c_n}\} \| c_m \| P\{x := 3\} \ | \ n[\{\mathsf{Int}^{\uparrow c_m}\} \| c_n \| \mathbf{0}]]. \end{array}$$

After this communication has been performed, x becomes 3 in P, and the input prefix $(x : \mathsf{Int})^{\downarrow c_n}$ and the output $\langle 3 \rangle^{\uparrow c_m}$ are consumed.

1.2. *Local interfaces*

As in other ambient calculi (Giovannetti 2003), different ambients in **BACI** may share the same ambient name, but, additionally, in **BACI** the same port name may also be used by different ambients. **BACI** not only allows an ambient to have any arbitrary port name associated with it, but also allows ambients with the same port name and the same ambient name to have different local views. For example, the two ambients

$$p[\,\{\mathsf{Int}^{\downarrow c_n}\} \parallel c_p \parallel (x : \mathsf{Int})^{\downarrow c_n}.P\,] \mid p[\,\{\mathsf{cap}^{\downarrow c_n}\} \parallel c_p \parallel (x : \mathsf{cap})^{\downarrow c_n}.Q\,]$$

both have the name p and port c_p, but differ in their local views.

Moreover, **BACI** can type the following example, which in statically typed calculi with global typing information would be rejected by the type-checker:

$$n[\{\mathsf{Int}^{\uparrow c_p}\} \parallel c_n \parallel \mathsf{in}\ p.\langle 3 \rangle^{\uparrow c_p}\,] \mid p[\{\mathsf{Int}^{\downarrow c_n}\} \parallel c_p \parallel (x : \mathsf{Int})^{\downarrow c_n}.P\,] \mid p[\{\mathsf{cap}^{\downarrow c_n}\} \parallel c_p \parallel (x : \mathsf{cap})^{\downarrow c_n}.Q\,]\,.$$

In this example, ambient n wants to enter ambient p; however, there are two different ambients called p and only one of them can receive the 3 that n wants to send. The reason this example would be rejected in other calculi is that ambients p have different types (one declaring that it can communicate an Int and the other declaring that it can communicate a capability, cap), contradicting the fact that names, such as p, have a unique type associated with them in a global environment. The local typing information in **BACI**'s ambients will allow n to enter only the ambient p that can receive the 3 that n wants to send.

In all the variants of ambient calculi considered in Giovannetti (2003), ambient n could enter either of the ambients named p, because those ambients would necessarily have the same type. However, since **BACI** allows different types for ambients with the same name, it only allows entry to the ambient that expects an Int, preventing a type mismatch during communication – we will illustrate this feature with an example in Subsection 3.2.

1.3. *Knowledge acquisition*

When an ambient enters another ambient, the host and the entering ambients can exchange their communication ports and establish a TOC between them. This is accomplished by sibling ambients using the actions $\mathsf{inC}(v{:}\,\tilde{\varphi})m$ and $\overline{\mathsf{inC}}(u : \tilde{\varphi})$, where m is the destination ambient, $\tilde{\varphi}$ is the topic of conversation, and the variables v and u are formal parameters for the communication ports of the destination and entering ambients. For example,

$$n[\,\varnothing \parallel c_n \parallel \mathsf{inC}(v{:}\,\mathsf{cap})m.\langle \mathsf{in}\ p \rangle^{\uparrow v}\,] \mid m[\,\varnothing \parallel c_m \parallel \overline{\mathsf{inC}}(u : \mathsf{cap}).(x : \mathsf{cap})^{\downarrow u}.x\,]$$
$$\longrightarrow$$
$$m[\,\{\mathsf{cap}^{\downarrow c_n}\} \parallel c_m \parallel n[\,\{\mathsf{cap}^{\uparrow c_m}\} \parallel c_n \parallel \langle \mathsf{in}\ p \rangle^{\uparrow c_m}\,] \mid (x : \mathsf{cap})^{\downarrow c_n}.x\,]$$
$$\longrightarrow$$
$$m[\,\{\mathsf{cap}^{\downarrow c_n}\} \parallel c_m \parallel n[\,\{\mathsf{cap}^{\uparrow c_m}\} \parallel c_n \parallel \mathbf{0}\,] \mid \mathsf{in}\ p\,]\,.$$

At first, both ambients have no knowledge about each other's ports because their local views are empty. However, when n enters m, the ambients exchange their ports and they replace the port variables v and u bound by inC and $\overline{\mathsf{inC}}$ with the actual port names c_m and c_n. During the exchange, the local views are also updated, reflecting the fact that the processes inside the ambients will communicate using those newly identified port names.

1.4. *Related work*

Modelling the world wide web requires a notion of process at a given location and a location space where processes can move from one location to another. In the earliest proposals, such as the *Dπ-calculus* (Hennessy and Riely 2002) and the language *Klaim* (De Nicola *et al.* 1998), the structure of locations is flat. The *Mobile Ambient (MA)* calculus (Cardelli and Gordon 2000) deals with a hierarchical structure of locations called ambients. An interesting *core model* generalising many of the available calculi and languages has been developed within the Mikado project (Boudol 2003).

Many variants of MA have been designed: see Giovannetti (2003) for a tutorial. A crucial choice to be made in all these calculi is the form of *interaction* between processes in different ambients. In the original calculus (Cardelli and Gordon 2000), interaction is only local to an ambient; therefore, in order for processes in different ambients to communicate, at least one of the ambients' boundaries has to be dissolved. In Cardelli and Gordon (2000), Amtoft *et al.* (2001), Bugliesi and Castagna (2002), Merro and Hennessy (2006) and Levi and Sangiorgi (2003), the open capability dissolves the ambient boundary.

The calculus **M3** (Coppo *et al.* 2003; Coppo *et al.* 2004) allows general process mobility. In *Boxed Ambients (BA)* (Merro and Sassone 2002; Bugliesi *et al.* 2004; Bugliesi *et al.* 2005), parents and children can communicate as in the *Seal* calculus (Castagna *et al.* 2005). Our calculus, **BACI**, follows this last protocol.

The *co-actions* (which were first introduced in Levi and Sangiorgi (2003), and then used with modifications in Bugliesi and Castagna (2002), Merro and Hennessy (2006), Merro and Sassone (2002) and Bugliesi *et al.* (2005)) require the agreement of the 'passive' ambients involved in mobility. The co-actions of **BACI**, in which port names are communicated, were inspired by those of Bugliesi *et al.* (2005), though there the communication only involves the name of the entering ambient.

Ambient calculi are often typed: the types assure behavioural properties concerning communication, mobility, resource access, security, and so on (Cardelli *et al.* 2002; Amtoft *et al.* 2001; Bugliesi and Castagna 2002; Merro and Hennessy 2006; Merro and Sassone 2002; Levi and Sangiorgi 2003; Barbanera *et al.* 2003; Bugliesi *et al.* 2004; Lhoussaine and Sassone 2004; Bugliesi *et al.* 2005). To our knowledge, before **BACI**, only the calculi of Hennessy and Riely (2003), Bugliesi and Castagna (2002), Coppo *et al.* (2004) and Coppo *et al.* (2005) consider *type information local to ambients*, while in the other proposals there is a global environment containing all typing assumptions. When dealing with computing in wide area 'open' systems it is sensible to assume the existence of different local environments. The price to pay is that static checks are no longer enough to assure correctness: we now need to carry typing information at run time. Following ideas from Goguen (1995), Goguen (1999) and Hennessy and Riely (2003), we define an *operational semantics with types*, which is simpler than a fully-fledged *typed operational semantics* in the sense that we only need to check agreement between the local views upon mobility. In both **BACI** and the calculus of Hennessy and Riely (2003) mobility is constrained by type-checking: the difference is that while in Hennessy and Riely (2003) a whole process must be type checked in order to see if it agrees with the view (called a *filter*) of the destination location, **BACI** only compares the types of the ports in the local views of the moving and destination ambients. To reduce the need for type-checking, Hennessy and Riely (2003) introduces the notion of *trust* between locations: processes originating at trusted locations need not be type checked.

In some sense, the run-time checking of **BACI** can be seen as a special case of *proof carrying code* (Necula 1997), since an ambient can be seen as mobile code that carries typing information to enable or disable mobility.

The local type information in **BACI** can dynamically increase with ambient movements: this is inspired by the mechanisms of knowledge acquisition for the Dπ-calculus considered in Hennessy *et al.* (2004).

The flexibility of sub-typing alone (see, for example, Zimmer (2000) and Merro and Sassone (2002)) does not allow different TOCs with different parents and with children sharing the same ambient name, or an increase in the type information.

Behavioural types (Amtoft *et al.* 2001; Amtoft *et al.* 2004) resemble computational traces in allowing polymorphic communications. **BACI**'s communication interfaces are also a permissive tool for typing non-local communications. In Amtoft *et al.* (2004), the type of communication with the parent changes when communication takes place. However, there is no named communication with the parent, making it impossible to express the fact that communication with different parents has different types, as in our last example.

In Section 4 we will discuss the behaviour of **BACI** processes by introducing a labelled transition system, which, essentially, follows Merro and Hennessy (2006), Coppo *et al.* (2003) and Bugliesi *et al.* (2005).

BACI was extended in Garralda and Compagnoni (2005) by the addition of multiple ports and port restriction. Furthermore, **BACI** provided the motivation for $\mathbf{BACI}_R$, a mobile ambient calculus with distributed role-based access control (Compagnoni and Gunter 2005). Finally, Garralda *et al.* (2006) presents **BASS**, which is an extension of **BACI** with safe sessions (Honda *et al.* 1998; Bonelli *et al.* 2005; Dezani-Ciancaglini *et al.* 2006).

1.5. *Organisation of the paper*

Section 2 introduces the syntax of the calculus, its operational semantics, and the type system. Section 3 discusses two extended examples highlighting the features of **BACI**. Section 4 studies a reduction barbed congruence and a labelled transition system (LTS). The bisimilarity induced by the LTS is shown to be sound with respect to the congruence,

Table 1. *Syntax of* ***BACI***

Basic types			
φ	$::=$	amb	ambient
	$\mid$	cap	capability
Located types			
τ	$::=$	$(\varphi_1, \ldots, \varphi_k)^\eta$	
Names			
α, β	$::=$	n	name constant
	$\mid$	x	name variable
(Co)-Capabilities			
C, D	$::=$	$\mathsf{in}\ \alpha$	enter
	$\mid$	$\mathsf{out}\ \alpha$	exit
	$\mid$	$\overline{\mathsf{in}}$	allow enter
	$\mid$	$\overline{\mathsf{out}}$	allow exit
	$\mid$	$C.D$	path
	$\mid$	x	capability variable
Prefixes			
π	$::=$	C	capabilities
	$\mid$	$(x_1 : \varphi_1, \ldots, x_k : \varphi_k)^\eta$	input
	$\mid$	$\langle M_1, \ldots, M_k \rangle^\eta$	output
	$\mid$	$\mathsf{inC}(v{:}\,\varphi_1, \ldots, \varphi_k)\alpha$	port enter
	$\mid$	$\mathsf{outC}(v{:}\,\varphi_1, \ldots, \varphi_k)\alpha$	port exit
	$\mid$	$\overline{\mathsf{inC}}(v : \varphi_1, \ldots, \varphi_k)$	allow port enter
	$\mid$	$\overline{\mathsf{outC}}(v : \varphi_1, \ldots, \varphi_k)$	allow port exit
Communication types			
ρ	$::=$	shh	no exchange
	$\mid$	$(\varphi_1, \ldots, \varphi_k)$	exchange tuple
Local view			
Γ	$::=$	$\varnothing$	empty
	$\mid$	Γ, τ	interface
Ports			
γ	$::=$	c	port constant
	$\mid$	v	port variable
Locations			
η	$::=$	$\uparrow\gamma$	parent port γ
	$\mid$	$\downarrow\gamma$	child port γ
	$\mid$	$\star$	local
Messages			
M, N	$::=$	α	name
	$\mid$	C	capability
Pre-processes			
P	$::=$	$\mathbf{0}$	nil process
	$\mid$	$\pi.P$	prefixing
	$\mid$	$P_1 \mid P_2$	composition
	$\mid$	$\alpha[\![\, \Gamma \parallel c \parallel P \,]\!]$	ambient
	$\mid$	$!\pi.P$	guarded replication
	$\mid$	$(\boldsymbol{\nu} n)P$	restriction

and some congruence laws are identified. Finally, we present conclusions and suggest further research.

Most of the technical proofs are included in the Appendices. An extended abstract of this work was presented at MFCS 2004 (Bonelli *et al*. 2004). We have improved the syntax with respect to that earlier presentation, and include all the technical proofs that were omitted from the earlier work due to space restrictions.

2. The calculus

2.1. *Syntax of* ***BACI***

The syntax for types and terms of **BACI** is given in Table 1.

Types

BACI has two basic types: the type amb of ambient names and the type cap of capabilities. Communication types are the types of information being exchanged: shh denotes the absence of communication (no information exchange), and $(\varphi_1, \ldots, \varphi_k)$ is the type of a tuple of messages. Located types are communication types decorated with locations, of the form $(\varphi_1, \ldots, \varphi_k)^\eta$, where the location η can be $\star$ for local communication, $\downarrow\gamma$ for communication with a child port, or $\uparrow\gamma$ for communication with a parent port.

Terms

We assume two disjoint denumerable sets of variables: one for ambient name, capability and message variables, which are ranged over by $x, y, z, \ldots$; the other for port variables, which are ranged over by $v, u, \ldots$. We use $m, n, o, p, q \ldots$ for ambient name constants and $x, y, z, \ldots$ for ambient name variables, and use α, β to range over both ambient constants and ambient variables. We use $\mathscr{N}$ for the set of ambient names. Communication port constants are written $c, c_n, \ldots$ and communication port variables are written $v, u, \ldots$, and we use γ to represent either a communication port constant or variable. The expressions $\mathsf{inC}(v\colon \varphi_1, \ldots, \varphi_k)\alpha$, $\mathsf{outC}(v\colon \varphi_1, \ldots, \varphi_k)\alpha$, $\overline{\mathsf{inC}}(v : \varphi_1, \ldots, \varphi_k)$, $\overline{\mathsf{outC}}(v : \varphi_1, \ldots, \varphi_k)$ are *binders* for v in prefixes and processes.

A pre-process is a *process* only if it is well formed according to the rules of Table 9. Hence, process and well-formed process are synonyms.

The process $\mathbf{0}$ is the null process, $P_1 \mid P_2$ denotes the parallel composition of processes P_1 and P_2, and $(\boldsymbol{\nu} n)P$ is the usual restriction operator that binds all free occurrences of n in P. The expression $\pi.P$ denotes the process that performs an action or a co-action π and then continues with P. The π action includes input/output (I/O) actions and mobility actions. The I/O exchanges are directed upwards to the parent ambient, downwards to a child ambient or locally to other processes at the same level. The direction of each communication is determined by the superscript η. Only prefixed processes can be replicated; as usual, the symbol ! denotes the replication operator. Replicated input in the π-calculus has the same expressive power of full replication (Honda and Yoshida 1994) and recursion (Milner 1993; Sangiorgi and Walker 2002). Moreover, prefixed replication allows a simpler labelled transition, that is, the transition (LTS REPL) of Table 15, as discussed in Section 4. Finally, note that the proof of congruence of full bisimilarity (Theorem 4.10) relies on this restriction.

Information is exchanged between processes by communicating tuples of messages. Each message can be either an ambient name or a (co-)capability[†]. The ambient names received as messages can substitute an ambient name variable in an ambient constructor or in a capability. Capabilities and co-capabilities constitute the mobility actions and co-actions: the capability $\mathsf{in}\ \alpha$ allows an ambient to enter ambient α, and the capability $\mathsf{out}\ \alpha$ allows an ambient to exit ambient α. In order to be executed, each capability must be matched at the destination ambient with a corresponding co-capability $\overline{\mathsf{in}}$ or $\overline{\mathsf{out}}$. Both capabilities and co-capabilities can be sent as messages. A single (co-)capability or several (co-)capabilities forming a path may be sent.

In addition to these standard mobility actions and co-actions, **BACI** introduces the inC and outC actions and their corresponding co-actions $\overline{\mathsf{inC}}$ and $\overline{\mathsf{outC}}$. These actions and co-actions are similar to the enter and exit (co-)capabilities. However, they also have a port variable, which is bound at execution time with the port of the counterpart ambient involved in the mobility action.

[†] It is easy to extend the types of messages to handle basic types such as integer or boolean without any technical problems, but we shall not do so here for the sake of simplicity.

Table 2. *Structural congruence*

$!P \equiv P \mid !P$	(Struct Rep Par)
$(\boldsymbol{\nu} n)(\boldsymbol{\nu} m)P \equiv (\boldsymbol{\nu} m)(\boldsymbol{\nu} n)P$	(Struct Res Res)
$(\boldsymbol{\nu} n)(P \mid Q) \equiv P \mid (\boldsymbol{\nu} n)Q$, if $n \notin \mathrm{fn}(P)$	(Struct Res Par)
$(\boldsymbol{\nu} n)m[\![\Gamma_m \parallel c_m \parallel P]\!] \equiv m[\![\Gamma_m \parallel c_m \parallel (\boldsymbol{\nu} n)P]\!]$, if $n \neq m$	(Struct Res Amb)
$(C.D).P \equiv C.D.P$	(Struct .)

Port names cannot be sent as messages; therefore, the only way of learning a port name is by using the inC and outC actions with their co-actions. In their execution, the ambients affected by this action exchange port names using the binders in these special (co-)actions. Additionally, in order to retain typability, port variables have an associated exchange tuple $(\varphi_1, \ldots, \varphi_k)$.

An ambient is written $\alpha[\![\Gamma \parallel c \parallel P]\!]$, where: α is an ambient name constant or an ambient name variable; Γ is the local view, a finite set of located types; c is the communication port; and P is the enclosed process.

Each local view Γ can be seen as a function from locations to exchange tuples, so a particular location cannot appear twice with different tuples in the same Γ.

A process is said to be *closed* if it does not contain any free variables. Processes differing only in the names of their bound variables are considered equal. In the following, we always consider well-formed processes unless we say explicitly otherwise.

Notation: We write $\tilde{\varphi}$ as a shorthand for $(\varphi_1, \ldots, \varphi_k)$ and $\tilde{x} : \tilde{\varphi}$ for $(x_1 : \varphi_1, \ldots, x_k : \varphi_k)$. If $\tilde{x} = (x_1, \ldots, x_n)$ and $\tilde{M} = (M_1, \ldots, M_n)$, we write $P\{\tilde{x} := \tilde{M}\}$ for the simultaneous capture-free substitution of all free occurrences of x_i by M_i, $1 \leqslant i \leqslant n$. We extend this notation to other syntactic constructs throughout the paper. We write $\mathrm{fn}(P)$ for the set of free ambient names in P, and $\mathrm{fv}(P)$ for the set of free variables in P.

2.2. *Operational semantics*

The operational semantics is defined in terms of structural congruence and reduction rules.

Structural congruence is the least congruence such that the set of processes is a commutative monoid with respect to $\mid$, having $\mathbf{0}$ as its neutral element, and the axioms of Table 2 are satisfied. This definition is standard and equates processes that should be regarded as essentially the same from an operational point of view.

The reduction relation is given by three groups of rules: *mobility*, *communication* and *structural*. The structural rules are standard. Before describing mobility and communication, we need the definitions presented in Tables 3 and 4.

The *application* $\Gamma(\eta)$ *of a local view* Γ *to a location* η returns the exchange tuple associated with η in Γ if there exists a corresponding located type in Γ. Otherwise, the application returns shh, which means that the location has no associated type (that is, the location does not occur in Γ).

The *extension* $\oplus$ *of a local view* Γ *with a located type* $\tilde{\varphi}^\eta$ has three possible outcomes:

— if η does not occur in Γ, it is the local view $\Gamma, \tilde{\varphi}^\eta$ obtained by adding $\tilde{\varphi}^\eta$ to Γ;

Table 3. *Operations on locations and local views*

Application of local views to locations

$$\Gamma(\eta) = \begin{cases} \tilde{\varphi} & \text{if } \tilde{\varphi}^{\eta} \in \Gamma, \\ \mathsf{shh} & \text{otherwise.} \end{cases}$$

Addition of located types to local views

$$\Gamma \oplus \tilde{\varphi}^{\eta} = \begin{cases} \Gamma, \tilde{\varphi}^{\eta} & \text{if } \Gamma(\eta) = \mathsf{shh}, \\ \Gamma & \text{if } \Gamma(\eta) = \tilde{\varphi}, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Preorder on communication types

$$\rho \preceq \rho' \text{ if, and only if } \rho = \rho' \text{ or } \rho = \mathsf{shh}$$

Table 4. *Location substitution on processes*

$\mathbf{0}\{\ddagger c/\ddagger v\}$	$=$	$\mathbf{0}$	
$(P_1 \mid P_2)\{\ddagger c/\ddagger v\}$	$=$	$P_1\{\ddagger c/\ddagger v\} \mid P_2\{\ddagger c/\ddagger v\}$	
$((\boldsymbol{\nu} n)P)\{\ddagger c/\ddagger v\}$	$=$	$(\boldsymbol{\nu} n)(P\{\ddagger c/\ddagger v\})$	
$(!P)\{\ddagger c/\ddagger v\}$	$=$	$!(P\{\ddagger c/\ddagger v\})$	
$\alpha[\Gamma_\alpha \parallel c_\alpha \parallel P]\{\ddagger c/\ddagger v\}$	$=$	$\alpha[\Gamma_\alpha \parallel c_\alpha \parallel P]$	
$((\tilde{x} : \tilde{\varphi})^{\eta}.P)\{\ddagger c/\ddagger v\}$	$=$	$(\tilde{x} : \tilde{\varphi})^{\eta}.(P\{\ddagger c/\ddagger v\})$	if $\eta \neq \ddagger v$
$((\tilde{x} : \tilde{\varphi})^{\ddagger v}.P)\{\ddagger c/\ddagger v\}$	$=$	$(\tilde{x} : \tilde{\varphi})^{\ddagger c}.(P\{\ddagger c/\ddagger v\})$	
$(\langle \tilde{M} \rangle^{\eta}.P)\{\ddagger c/\ddagger v\}$	$=$	$\langle \tilde{M} \rangle^{\eta}.(P\{\ddagger c/\ddagger v\})$	if $\eta \neq \ddagger v$
$(\langle \tilde{M} \rangle^{\ddagger v}.P)\{\ddagger c/\ddagger v\}$	$=$	$\langle \tilde{M} \rangle^{\ddagger c}.(P\{\ddagger c/\ddagger v\})$	
$(C.P)\{\ddagger c/\ddagger v\}$	$=$	$C.(P\{\ddagger c/\ddagger v\})$	
$(\mathsf{in/outC}(u{:}\,\tilde{\varphi})\alpha.P)\{\ddagger c/\ddagger v\}$	$=$	$\mathsf{in/outC}(u{:}\,\tilde{\varphi})\alpha.(P\{\ddagger c/\ddagger v\})$	if $u \neq v$
$(\overline{\mathsf{in/outC}}(u : \tilde{\varphi}).P)\{\ddagger c/\ddagger v\}$	$=$	$\overline{\mathsf{in/outC}}(u : \tilde{\varphi}).(P\{\ddagger c/\ddagger v\})$	if $u \neq v$

— if Γ does already contain $\tilde{\varphi}^{\eta}$, then it is simply Γ; and
— it is undefined otherwise.

The set of communication types with the preorder $\preceq$ is the flat domain whose bottom is shh.

Let $\ddagger \in \{\uparrow, \downarrow\}$. The *location substitution on processes* $P\{\ddagger c/\ddagger v\}$ recursively replaces free occurrences of the location $\ddagger v$ with the location $\ddagger c$ in all input and output prefixes except across ambient boundaries. This requirement justifies the use of a different notation for location substitutions compared with that used for message substitutions. We use the convention of renaming the port variables bound in prefixes to ensure capture-free substitutions. Note that $\Gamma(\eta)$ and $P\{\ddagger c/\ddagger v\}$ are always defined, while $\Gamma \oplus \tilde{\varphi}^{\eta}$ may be undefined.

The *communication rules* in Table 5 include three different message passing forms:

— *local* between two processes inside the same ambient,
— *input* from a process inside an ambient, and
— *output* to a process inside an ambient.

These rules are fairly standard; however, instead of using the ambient names to establish communication, the processes involved in the communication use port names. In order to engage in communication with a process at the immediate upper level, a process needs

Table 5. *Operational semantics (communication)*

(RED-LOCAL)

$$(\tilde{x} : \tilde{\varphi})^{\star}.P \mid \langle \tilde{M} \rangle^{\star}.Q \quad \longrightarrow \quad P\{\tilde{x} := \tilde{M}\} \mid Q$$

(RED-RECV↓)

$$m[\,\Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi})^{\downarrow c_n}.P \mid n[\,\Gamma_n \parallel c_n \parallel \langle \tilde{M} \rangle^{\uparrow c_m}.Q \mid R\,] \mid S\,]$$
$$\longrightarrow$$
$$m[\,\Gamma_m \parallel c_m \parallel P\{\tilde{x} := \tilde{M}\} \mid n[\,\Gamma_n \parallel c_n \parallel Q \mid R\,] \mid S\,]$$

(RED-SEND↓)

$$m[\,\Gamma_m \parallel c_m \parallel \langle \tilde{M} \rangle^{\downarrow c_n}.P \mid n[\,\Gamma_n \parallel c_n \parallel (\tilde{x} : \tilde{\varphi})^{\uparrow c_m}.Q \mid R\,] \mid S\,]$$
$$\longrightarrow$$
$$m[\,\Gamma_m \parallel c_m \parallel P \mid n[\,\Gamma \parallel c_n \parallel Q\{\tilde{x} := \tilde{M}\} \mid R\,] \mid S\,]$$

Table 6. *Operational semantics (mobility)*

(RED-ENTER)

$$n[\,\Gamma_n \parallel c_n \parallel \mathsf{in}\ m.P_1 \mid P_2\,] \mid m[\,\Gamma_m \parallel c_m \parallel \overline{\mathsf{in}}.Q_1 \mid Q_2\,]$$
$$\longrightarrow$$
$$m[\,\Gamma_m \parallel c_m \parallel n[\,\Gamma_n \parallel c_n \parallel P_1 \mid P_2\,] \mid Q_1 \mid Q_2\,]$$
$$\text{if } \Gamma_n(\uparrow c_m) \preceq \Gamma_m(\downarrow c_n)$$

(RED-EXIT)

$$p[\,\Gamma_p \parallel c_p \parallel n[\,\Gamma_n \parallel c_n \parallel m[\,\Gamma_m \parallel c_m \parallel \mathsf{out}\ n.P_1 \mid P_2\,] \mid Q\,] \mid \overline{\mathsf{out}}.R_1 \mid R_2\,]$$
$$\longrightarrow$$
$$p[\,\Gamma_p \parallel c_p \parallel m[\,\Gamma_m \parallel c_m \parallel P_1 \mid P_2\,] \mid n[\,\Gamma_n \parallel c_n \parallel Q\,] \mid R_1 \mid R_2\,]$$
$$\text{if } \Gamma_m(\uparrow c_p) \preceq \Gamma_p(\downarrow c_m)$$

(RED-ENTERC)

$$n[\,\Gamma_n \parallel c_n \parallel \mathsf{inC}(v : \tilde{\varphi})m.P_1 \mid P_2\,] \mid m[\,\Gamma_m \parallel c_m \parallel \overline{\mathsf{inC}}(u : \tilde{\varphi}).Q_1 \mid Q_2\,]$$
$$\longrightarrow$$
$$m[\,\Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n} \parallel c_m \parallel n[\,\Gamma_n \oplus \tilde{\varphi}^{\uparrow c_m} \parallel c_n \parallel P_1\{\uparrow c_m/\uparrow v\} \mid P_2\,] \mid Q_1\{\downarrow c_n/\downarrow u\} \mid Q_2\,]$$
$$\text{if } \Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n} \text{ and } \Gamma_n \oplus \tilde{\varphi}^{\uparrow c_m} \text{are defined}$$

(RED-EXITC)

$$p[\,\Gamma_p \parallel c_p \parallel n[\,\Gamma_n \parallel c_n \parallel m[\,\Gamma_m \parallel c_m \parallel \mathsf{outC}(v : \tilde{\varphi})n.P_1 \mid P_2\,] \mid Q\,] \mid \overline{\mathsf{outC}}(u : \tilde{\varphi}).R_1 \mid R_2\,]$$
$$\longrightarrow$$
$$p[\,\Gamma_p \oplus \tilde{\varphi}^{\downarrow c_m} \parallel c_p \parallel m[\,\Gamma_m \oplus \tilde{\varphi}^{\uparrow c_p} \parallel c_m \parallel P_1\{\uparrow c_p/\uparrow v\} \mid P_2\,] \mid n[\,\Gamma_n \parallel c_n \parallel Q\,] \mid R_1\{\downarrow c_m/\downarrow u\} \mid R_2\,]$$
$$\text{if } \Gamma_p \oplus \tilde{\varphi}^{\downarrow c_m} \text{ and } \Gamma_m \oplus \tilde{\varphi}^{\uparrow c_p} \text{are defined}$$

to use the port name associated with its parent ambient. Similarly, if a process wants to communicate with another process inside an ambient, it needs to use the port name associated with the ambient.

As shown in Table 6, the *mobility rules* consist of two pairs of rules: the rules that exercise the simple entry and exit capabilities, and the rules that use special primitives, which are similar to capabilities, but which additionally establish a TOC and exchange the port names associated with the ambients involved in the movement.

Table 7. *Operational semantics (structural)*

(RED-STRUCT)	$\dfrac{P \equiv P', \quad P' \longrightarrow Q', \quad Q' \equiv Q}{P \longrightarrow Q}$		
(RED-PAR)	$P \longrightarrow Q$	$\Longrightarrow$	$P \mid R \longrightarrow Q \mid R$
(RED-RES)	$P \longrightarrow Q$	$\Longrightarrow$	$(\boldsymbol{\nu} n)P \longrightarrow (\boldsymbol{\nu} n)Q$
(RED-AMB)	$P \longrightarrow Q$	$\Longrightarrow$	$n[\![\Gamma \parallel c \parallel P]\!] \longrightarrow n[\![\Gamma \parallel c \parallel Q]\!]$

The rules (RED-ENTER) and (RED-EXIT) require that the local views of the moving and destination ambients agree. This condition prevents type mismatch during possible message exchanges, as we will show at the end of the present section.

Consider (RED-ENTER), for instance. If the entering ambient n is willing to communicate with the host ambient m using a type $\tilde{\varphi}$, the restriction $\Gamma_n(\uparrow c_m) \preceq \Gamma_m(\downarrow c_n)$ requires that the processes inside m also use the type $\tilde{\varphi}$ when they are communicating with the location $\downarrow c_n$. This restriction allows some flexibility: if the incoming ambient is silent with respect to the communication with the host ambient, no restriction is really imposed on the local view of the host ambient. In that case, there is no risk of type mismatch in future message exchanges.

Note that the rule (RED-EXIT) includes the outermost ambient p, which is the destination for the moving ambient m. Including ambient p is necessary in order to compare the local views of the ambients m and p. The destination ambient does not appear in the original formulation (Bugliesi *et al.* 2004), but it does contain the exiting co-action in the calculi of Merro and Hennessy (2006) and Teller *et al.* (2002).

After an ambient moves to a new location, the ambient can start communicating with processes at the host ambient. However, the processes at both ends of the communication must have prior knowledge of the communication port associated with that location.

In order to enable a process to learn dynamically the port name associated with the location it is moving to, we introduce the inC, outC actions and the $\overline{\mathsf{inC}}$, $\overline{\mathsf{outC}}$ co-actions. As mentioned earlier, these action prefixes bind port variables. In both the rules (RED-ENTERC) and (RED-EXITC), the port names of the ambients involved in the movement are exchanged: the moving ambient gets the port name associated with the destination and the ambient that receives the moving ambient obtains the port associated with it. In this way, the port variables bound by these prefixes are substituted by the corresponding port names within the continuation processes.

The variables bound in these actions and co-actions have a communication type $\tilde{\varphi}$. This is the type that the continuation process is expecting on that port. Therefore, to prevent type mismatches during communication, the types of the actions and co-actions are required to be the same. After the reduction, the type of the process inside the ambient affected by the movement may change, since new information was given to their processes. Consequently, the local views need to be updated as well. The resulting local views must not have any conflicting types. Therefore, we include a side condition in the (RED-ENTERC) and (RED-EXITC) rules to guarantee that updated local views are defined.

Table 8. *Well-formed Messages*

$$\text{(amb)}\ \frac{n \in \mathcal{N}}{\Sigma \vdash n : \mathsf{amb}} \qquad \text{(VAR)}\ \frac{}{\Sigma, x:\varphi \vdash x : \varphi} \qquad \text{(cocap)}\ \frac{C \in \{\overline{\mathsf{in}}, \overline{\mathsf{out}}\}}{\Sigma \vdash C : \mathsf{cap}}$$

$$\text{(cap)}\ \frac{\Sigma \vdash \alpha : \mathsf{amb} \quad C \in \{\mathsf{in}\ \alpha, \mathsf{out}\ \alpha\}}{\Sigma \vdash C : \mathsf{cap}} \qquad \text{(cap}-\text{COMP)}\ \frac{\Sigma \vdash C : \mathsf{cap} \quad \Sigma \vdash D : \mathsf{cap}}{\Sigma \vdash C.D : \mathsf{cap}}$$

2.3. *Typing rules*

A typing environment, denoted by Σ, declares variables of type amb and cap.

$$\begin{array}{rcll} \Sigma & ::= & \varnothing & \text{empty environment} \\ & | & \Sigma, x : \varphi & \text{variable declaration} \end{array}$$

In the rest of the paper, we only consider typing environments that assign a unique type to each variable in their domains. The typing rules define two judgements:

— $\Sigma \vdash M : \varphi$, read '$M$ is a well-formed message of type φ'; and
— $\Sigma \vdash_c P : \Gamma$, read '$P$ is a well-formed process, assuming the local communication interface of its host consists of the communication port c and of the local view Γ'.

The typing rules for the first judgement are given in Table 8.

The (amb) rule assigns to an ambient name the constant type amb rather than a more informative type, as in the majority of systems (Giovannetti 2003). Indeed, more informative types presuppose the availability of global information on the type of ambients. In our setting, which is based on local views, the only assumption we make is that we can identify an ambient name when we see one.

The (cap) and (cocap) rules are also simpler than in formulations based on global knowledge of the communication types of ambients, since the control of message exchanges enabled by movements is delegated to run time.

The (VAR) and (cap—COMP) rules are standard.

The rules defining the judgement $\Sigma \vdash_c P : \Gamma$ are given in Table 9.

The typing rule (INACT) says that, since $\mathbf{0}$ does not interact with its host, it may be typed under a communication interface consisting of any port name c and any local view Γ.

The rule for replication (REP) is standard; however, the rule (RES) is not. Normally, the name n together with its type is assumed to belong to the global context Σ, where it is associated with a type expressing properties of processes that stay in ambients named n. However, in our local setting, the only relevant information is that n is an ambient name, that is, $n \in \mathcal{N}$.

The rule for parallel composition (COMP) is also standard.

Table 9. *Well-formed processes*

$$\text{(INACT)}\ \frac{}{\Sigma \vdash_c \mathbf{0} : \Gamma} \qquad \text{(REP)}\ \frac{\Sigma \vdash_c P : \Gamma}{\Sigma \vdash_c\ !P : \Gamma} \qquad \text{(RES)}\ \frac{\Sigma \vdash_c P : \Gamma \quad n \in \mathcal{N}}{\Sigma \vdash_c (\boldsymbol{\nu} n)P : \Gamma}$$

$$\text{(COMP)}\ \frac{\Sigma \vdash_c P_1 : \Gamma \quad \Sigma \vdash_c P_2 : \Gamma}{\Sigma \vdash_c P_1 \mid P_2 : \Gamma} \qquad \text{(CAP)}\ \frac{\Sigma \vdash C : \mathsf{cap} \quad \Sigma \vdash_c P : \Gamma}{\Sigma \vdash_c C.P : \Gamma}$$

$$\text{(CAPC)}\ \frac{\Sigma \vdash_c P : \Gamma, \tilde{\varphi}^{\Uparrow v} \quad \pi \in \{\mathsf{inC}(v{:}\,\tilde{\varphi})\alpha, \mathsf{outC}(v{:}\,\tilde{\varphi})\alpha\} \quad \Sigma \vdash \alpha : \mathsf{amb}}{\Sigma \vdash_c \pi.P : \Gamma}$$

$$\text{(COCAPC)}\ \frac{\Sigma \vdash_c P : \Gamma, \tilde{\varphi}^{\Downarrow v} \quad \pi \in \{\overline{\mathsf{inC}}(v : \tilde{\varphi}), \overline{\mathsf{outC}}(v : \tilde{\varphi})\}}{\Sigma \vdash_c \pi.P : \Gamma}$$

$$\text{(RECV)}\ \frac{\Sigma, x_1 : \varphi_1, \ldots, x_k : \varphi_k \vdash_c P : \Gamma \quad \Gamma(\eta) = (\varphi_1, \ldots, \varphi_k)}{\Sigma \vdash_c (x_1 : \varphi_1, \ldots, x_k : \varphi_k)^{\eta}.P : \Gamma}$$

$$\text{(SEND)}\ \frac{\Sigma \vdash_c P : \Gamma \quad \Sigma \vdash M_i : \varphi_i\ (1 \leqslant i \leqslant k) \quad \Gamma(\eta) = (\varphi_1, \ldots, \varphi_k)}{\Sigma \vdash_c \langle M_1, \ldots, M_k \rangle^{\eta}.P : \Gamma}$$

$$\text{(AMB)}\ \frac{\Sigma \vdash_{c'} P : \Gamma' \quad \Sigma \vdash \alpha : \mathsf{amb} \quad \Gamma'(\uparrow c) \preceq \Gamma(\downarrow c') \quad \Gamma' \text{ is closed}}{\Sigma \vdash_c \alpha[\,\Gamma' \parallel c' \parallel P\,] : \Gamma}$$

The typing rule (CAP) says that the only information given by capabilities is the fact that they are capabilities. Since we rely purely on local information, we shall relegate the correct use of capabilities to run time.

A process of the form $\mathsf{inC}(v{:}\,\rho)\alpha.P$ is well formed under the assumption that the host ambient has local view Γ only if P is well formed under the assumption that the host ambient has local view $\Gamma, \tilde{\varphi}^{\Uparrow v}$. In this way, we require that P is typed with a local view where the parent port v has the type $\tilde{\varphi}$ associated with it.

The typing of the other prefixes mentioned in rules (CAPC) and (COCAPC) is similar. The difference between these two rules is that in the first rule the process communicates with a new host ambient, while in the second it communicates with a newly entering child ambient.

The rules (Recv) and (Send) require that the type of the information that is exchanged together with its location must belong to the local view of the host ambient.

The (Amb) rule may be interpreted as follows. In order for $\alpha[\![\,\Gamma' \parallel c' \parallel P\,]\!]$ to be considered a well-formed process under a host ambient whose communication interface consists of a port c and a local view Γ, it must be the case that:

1 process P is well formed under a host ambient whose communication interface consists of port c' and local view Γ';
2 α must be an ambient name or an ambient variable;
3 either $\alpha[\![\,\Gamma' \parallel c' \parallel P\,]\!]$ communicates with its host ambient using the same communication type, or the ambient α does not engage in any communication with the host ambient (condition $\Gamma'(\uparrow c) \preceq \Gamma(\downarrow c')$);
4 no free port variables should occur in Γ', that is, Γ' should be *closed*.

The condition $\Gamma'(\uparrow c) \preceq \Gamma(\downarrow c')$ guarantees that, in the local view of the host ambient, there is a record of the communication types that child ambients use to communicate with the host. This constraint must be enforced regardless of whether or not the processes at the host ambient effectively communicate using the ports associated with the child ambient. Observe that port variables on the processes inside ambients may be replaced eventually by actual port names and the local view of the ambients will be updated accordingly. If the local views of host ambients do not record all of the information about the communication types used by any child ambient, some conflicts may arise after the substitution of a port variable with a port name.

Consider the following example. If the condition in question is not enforced, the pre-process (which is not a process!)

$$m[\![\,\varnothing \parallel c \parallel \overline{\mathsf{inC}}(v_1 : \mathsf{amb}).\langle q\rangle^{\downarrow v_1} \mid n[\![\,\{\mathsf{cap}^{\uparrow c}\} \parallel c' \parallel (x : \mathsf{cap})^{\uparrow c}.x\,]\!]$$

could be well formed for any port, local view and typing assumptions. Notice that the process in m does not use the location $\downarrow c'$: it is silent with respect to this port. However, the child ambient n is willing to communicate with its current host exchanging messages of type cap.

This pre-process is fine on its own. However, remember that the local views can be altered after the application of the (Red-EnterC) or (Red-ExitC) rules. If we put the previous process in parallel with the process

$$p[\![\,\varnothing \parallel c' \parallel \mathsf{inC}(v_2 : \mathsf{amb})m.(x : \mathsf{amb})^{\uparrow v_2}\,]\!]\,,$$

we can apply the rule (Red-EnterC) allowing the ambient p to enter ambient m resulting in

$$m[\![\,\{\mathsf{amb}^{\downarrow c'}\} \parallel c \parallel \langle q\rangle^{\downarrow c'} \mid p[\![\,\{\mathsf{amb}^{\uparrow c}\} \parallel c' \parallel (x : \mathsf{amb})^{\uparrow c}\,]\!] \mid n[\![\,\{\mathsf{cap}^{\uparrow c}\} \parallel c' \parallel (x : \mathsf{cap})^{\uparrow c}.x\,]\!]\,]\!]\,.$$

Note that the pre-process inside m can now communicate with either p or n. This particular situation has arisen because the entering ambient p has the port c' associated with it, which is the same port associated with the ambient n already inside m.

If we do not annotate the local view of m with the communication type used by its child ambient, we get the following expression, which is not a pre-process:

$$m[\,\{\mathsf{amb}^{\downarrow c'}\} \parallel c \parallel p[\,\{\mathsf{amb}^{\uparrow c}\} \parallel c' \parallel (x : \mathsf{amb})^{\uparrow c}\,] \mid n[\,\{\mathsf{cap}^{\uparrow c}\} \parallel c' \parallel q\,]\,].$$

This arises because the name q can be wrongly communicated to ambient n after applying rule (Red-Enter C) in the preprocess where a capability was expected.

The final condition for (Amb) requires that no port variables occur free in the local view of an ambient. This condition agrees with the definition of location substitution on processes (see Table 4). In fact, since the location substitution does not affect local views inside nested ambients, a free variable would always remain free. Allowing free variables in local views and propagating the location substitution inside nested ambients would lead to inconsistencies. For instance, in the pre-process

$$m[\,\varnothing \parallel c \parallel \mathsf{inC}(v\!:\!\mathsf{amb})n.P\,] \mid n[\,\varnothing \parallel c_n \parallel \overline{\mathsf{inC}}(u : \mathsf{amb}).(p[\,\{\mathsf{cap}^{\downarrow u}, \mathsf{amb}^{\downarrow c}\} \parallel c' \parallel Q\,])\,],$$

the ambient m could enter ambient n thanks to the (Red-EnterC) rule (see Table 6), which replaces the port variables in both ambients by the actual port names. The resulting process would be

$$n[\,\Gamma_n \parallel c_n \parallel m[\,\Gamma_m \parallel c \parallel P\{\uparrow c_n/\uparrow v\}\,] \mid p[\,\{\mathsf{cap}^{\downarrow c}, \mathsf{amb}^{\downarrow c}\} \parallel c' \parallel Q\{\downarrow c/\downarrow u\}\,]\,]$$

where $\Gamma_n = \{\mathsf{amb}^{\downarrow c}\}$, $\Gamma_m = \{\mathsf{amb}^{\uparrow c_n}\}$, and the local view of the ambient p is inconsistent.

The type system guarantees that communication inside ambients and across ambient boundaries never leads to type mismatches. This is formalised in the following theorem.

Theorem 2.1 (Subject Reduction). If $\Sigma \vdash_c P : \Gamma$ and $P \longrightarrow Q$, then $\Sigma \vdash_c Q : \Gamma$.

The proof of this theorem and supporting lemmas are included in Appendix A.

Finally, we want to prove formally that communication is safe, that is, no type mismatch can occur during communication. To this end, we extend without modification the reduction of processes to include pre-processes. Moreover, we add the constant *Error* to the syntax of pre-processes, and add the reduction rule

$$P \longrightarrow \mathit{Error}$$

when one of the following conditions holds:

— $P = (\tilde{x} : \tilde{\varphi})^\star.Q \mid \langle\tilde{M}\rangle^\star.R$, and $\varnothing \nvdash \tilde{M} : \tilde{\varphi}$;
— $P = m[\,\Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi})^{\downarrow c_n}.Q \mid n[\,\Gamma_n \parallel c_n \parallel \langle\tilde{M}\rangle^{\uparrow c_m}.R \mid S\,] \mid T\,]$, and $\varnothing \nvdash \tilde{M} : \tilde{\varphi}$;
— $P = m[\,\Gamma_m \parallel c_m \parallel \langle\tilde{M}\rangle^{\downarrow c_n}.Q \mid n[\,\Gamma_n \parallel c_n \parallel (\tilde{x} : \tilde{\varphi})^{\uparrow c_m}.R \mid S\,] \mid T\,]$, and $\varnothing \nvdash \tilde{M} : \tilde{\varphi}$;
— $P = m[\,\Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi})^{\eta}.Q \mid S\,]$, and $\tilde{\varphi} \neq \Gamma_m(\eta)$;
— $P = m[\,\Gamma_m \parallel c_m \parallel \langle\tilde{M}\rangle^{\eta}.Q \mid S\,]$, and $\varnothing \nvdash \tilde{M} : \Gamma_m(\eta)$;
— $P = m[\,\Gamma_m \parallel c_m \parallel n[\,\Gamma_n \parallel c_n \parallel Q\,] \mid R\,]$, and $\Gamma_n(\uparrow c_m) \not\lesssim \Gamma_m(\downarrow c_n)$.

From the Subject Reduction Theorem we immediately get the following proposition.

Proposition 2.1. If P is a well-formed process and $P \longrightarrow Q$, then Q does not contain *Error*.

Clearly, for an arbitrary pre-process generated using the syntax of Table 1, it is easy to check if it is well formed according to the typing rules of Tables 8 and 9. In other words, there is no problem in writing a *type checking* algorithm for our calculus.

We think that it would not be sensible to infer the local views, since they express the communication policies of ambients. For this reason, we do not see how to design a reasonable *type inference* strategy for **BACI**.

3. Examples

In this section we sketch some examples to show the expressiveness of **BACI**. Before doing so, we define the following auxiliary notation to make the examples easier to read:

$$\alpha^{\rightleftharpoons}[\,\Gamma \parallel c \parallel P\,] \triangleq \alpha[\,\Gamma \parallel c \parallel !\overline{\mathsf{in}} \mid !\overline{\mathsf{out}} \mid P\,].$$

This allows sibling and nested ambients of α to freely enter and exit. Note that $\alpha^{\rightleftharpoons}$ allows the entry of ambients that do not communicate with α and of ambients whose communication port name is already known by $\alpha^{\rightleftharpoons}$. For the sake of clarity, we may omit the type of input variables, since they can be obtained from the context.

3.1. *Remote printer*

In this example we represent two networks using ambients $n1$ and $n2$. Suppose the client is located in network $n1$ and the printer in $n2$, and routing from network $n1$ to $n2$ is also required. For simplicity, we place $n1$ and $n2$ at the same nesting level inside a larger ambient, called *inter*. However, the locations $n1$ and $n2$ may be far from each other within the nesting hierarchies.

$$INTERNET \triangleq inter^{\rightleftharpoons}[\,\varnothing \parallel c \parallel N1 \mid N2\,]$$

$$N1 \triangleq n1^{\rightleftharpoons}[\,\varnothing \parallel c_1 \parallel CLIENT \mid ROUTER\,]$$

$$N2 \triangleq n2^{\rightleftharpoons}[\,\varnothing \parallel c_2 \parallel PRINTER\,].$$

The idea is that the client sends a print *job* to *PRINTER* via *ROUTER*. A *job* ambient should receive two parameters, data and printer name, from *CLIENT* after releasing the job. After receiving the parameters, the job exits the client and enters *ROUTER*. There, it receives the path to $n2$, where the printer is located. After reaching $n2$, the job enters the printer and communicates the data to be printed:

$$JOB_{cl} \triangleq job[\,\Gamma_{job} \parallel c_j \parallel (d,p)^{\uparrow c_{cl}}.\mathsf{out}\ cl.\mathsf{in}\ r1to2.(route)^{\uparrow c_r}.route.\mathsf{in}\ p.\langle d\rangle^{\uparrow c_{pr}}\,]$$

where $\Gamma_{job} \triangleq \{(\mathsf{data},\mathsf{amb})^{\uparrow c_{cl}}, \mathsf{cap}^{\uparrow c_r}, \mathsf{data}^{\uparrow c_{pr}}\}$.

Notice that the *job* ambient is able to communicate with different parent ports using different TOCs. Here, c_j is the job's port, c_{cl} is the client's port, c_r is the router's port and c_{pr} is the printer's port.

CLIENT spawns the job and sends the data to be printed using the *job* ambient. Then, the job is received by *ROUTER*, which communicates the route for reaching $n2$ to the job. Finally, the job enters *PRINTER* and delivers its data:

$$CLIENT \triangleq client1^{\rightleftharpoons}[\,\{(\mathsf{data},\mathsf{amb})^{\downarrow c_j}\} \parallel c_{cl} \parallel \langle(d1, printer1)\rangle^{\downarrow c_j} \mid\ !\ JOB_{client1}\,]$$

$$ROUTER \triangleq r1to2^{\rightleftharpoons}[\,\{\mathsf{cap}^{\downarrow c_j}\} \parallel c_r \parallel !\langle(\mathsf{out}\ r1to2.\mathsf{out}\ n1.\mathsf{in}\ n2)\rangle^{\downarrow c_j}\,]$$

$$PRINTER \triangleq printer1^{\rightleftharpoons}[\,\{\mathsf{data}^{\downarrow c_j}\} \parallel c_{pr} \parallel !\,(d)^{\downarrow c_j}\,].$$

Once it has delivered its data, the *job* ambient becomes inactive and useless. Using the algebraic properties of Section 4.3, we can show (using $\cong$ to denote barbed congruence as defined in Definition 4.3) that

$$job[\,\Gamma_{job} \parallel c_j \parallel \mathbf{0}\,] \cong \mathbf{0}\,,$$

and hence also that

$$printer1^{\rightleftharpoons}[\,\{\mathsf{data}^{\downarrow c_j}\} \parallel c_{pr} \parallel !\,(d:\mathsf{data})^{\downarrow c_j} \mid job[\,\Gamma_{job} \parallel c_j \parallel \mathbf{0}\,]\,] \cong$$

$$printer1^{\rightleftharpoons}[\,\{\mathsf{data}^{\downarrow c_j}\} \parallel c_{pr} \parallel !\,(d:\mathsf{data})^{\downarrow c_j}\,].$$

Then, all the ‘garbage’ ambients that accumulate inside the printer ambient can safely be discarded.

3.2. *Printer services*

We assume now that the printer can accept different file formats: for instance, *postscript* and *proprietary* formats. For each of these formats, we shall have a different type: we shall use ps to denote postscript format and prop to denote proprietary format.

The printer should have two different threads: one for managing the requests in postscript format and the other for managing the requests in proprietary format. After receiving a request, each thread translates it to an internal format, denoted by the type inter, which is understood by the printer drivers. After the printer receives data in inter format from one if its threads, it proceeds to print that data:

$$PRINTER \triangleq printer^{\rightleftharpoons}[\,\{\mathsf{inter}^{\downarrow c_{pr}}\} \parallel c_{int} \parallel !\,(d:\mathsf{inter})^{\downarrow c_{pr}} \mid PSTHREAD \mid PROPTHREAD\,]$$

$$PSTHREAD \triangleq thread[\,\{\mathsf{inter}^{\uparrow c_j}, \mathsf{ps}^{\downarrow c_j}\} \parallel c_{pr} \parallel !\ \overline{\mathsf{in}}.(x:\mathsf{ps})^{\downarrow c_j}.\langle\ ps2inter(x)\rangle^{\uparrow c_{int}}\,]$$

$$PROPTHREAD \triangleq thread[\,\{\mathsf{inter}^{\uparrow c_j}, \mathsf{prop}^{\downarrow c_j}\} \parallel c_{pr} \parallel !\ \overline{\mathsf{in}}.(x:\mathsf{prop})^{\downarrow c_j}.\langle\ prop2inter(x)\rangle^{\uparrow c_{int}}\,].$$

Note that the only difference between the two threads is the kind of data they receive from the printing job. This must be reflected in the local view of each thread. The TOC between a printing job (that is, port c_j) and the thread is either ps or prop depending on the thread function.

Similarly, we shall allow two kinds of printing jobs: one for each format. For instance, a postscript printing job would look like the process

$$PSJOB\ \triangleq job[\,\{\mathsf{ps}^{\uparrow c_{pr}}\} \parallel c_j \parallel \mathsf{in}\ printer.\mathsf{in}\ thread.\langle \mathrm{data}\rangle^{\uparrow c_{pr}}\,].$$

As with the threads in the printer, the two versions of printing jobs only differ in their local views (and the type of data they are carrying). Also, the mobility path followed by a job to reach a thread is the same. The typing information in the local views only allows jobs with the same data format to enter a thread, that is, only postscript jobs are allowed to enter the postscript thread and only jobs with data in the proprietary format are allowed to enter the thread that specifically handles that format.

Clearly, encoding a similar example in the original BA calculus (Bugliesi *et al.* 2004) would be rather cumbersome.

3.3. *File server cluster*

This example represents some free download sites in which the user has a list of servers to choose for his download. We require that every time a customer requests a file download, the cluster designates one server from all the available servers in the cluster (that is, all the servers that are not serving other clients) to serve that request. Additionally, we want a cluster administrator to be able to execute some administrative operations like shutting down or powering up any particular server. For this reason, we assign a unique and distinctive name to each server. However, we use a common port name and interface for all of them to allow the cluster to communicate with all of them:

$$CLUSTER \triangleq cluster^{\rightleftharpoons}[\,\Gamma_{clu} \parallel c_{clu} \parallel LOAD\_BAL \mid SRV1 \mid SRV2\,]$$
$$\text{where } \Gamma_{clu} \triangleq \{(\mathsf{amb,filename})^{\downarrow c_{srv}}\}$$

$$LOAD\_BAL \triangleq !(\overline{\mathsf{inC}}(v_{cl} : (\mathsf{amb,filename})).(clname, fn)^{\downarrow v_{cl}}.\langle clname, fn\rangle^{\downarrow c_{srv}}).$$

The *cluster* includes all the servers and the *load balancing* mechanism. This mechanism allows a client to enter the cluster: the cluster receives the client's request that it forwards to any available server. Note that the cluster does not know the client's communication port in advance, and *vice versa*: they are learnt on the ENTER reduction, where the port names replace the variables bound by inC and $\overline{\mathsf{inC}}$. Each server has two main sub-processes: the service itself and the *power management* process. The *SERVE* process receives the forwarded request from the cluster ambient, and then it responds by spawning a messenger ambient called *job*. This job reaches the client and delivers the requested file. Note that before receiving a request, *SERVE* waits for an 'on' message from the power management ambient called *pwr*. The *pwr* ambient is used to inform the serving process that the server is still on. We now show how to use this feature to 'shut down' a server:

$$SRVi \triangleq srvi^{\rightleftharpoons}[\,\Gamma_{srvi} \parallel c_{srv} \parallel !(on)^{\downarrow c_{pwr}}.SERVE \mid PWR\,]$$
$$\text{where } \Gamma_{srvi} \triangleq \{\mathsf{onMsg}^{\downarrow c_{pwr}}, (\mathsf{amb,filename})^{\uparrow c_{clu}}\}$$

$$SERVE \triangleq (clname, fname)^{\uparrow c_{clu}}.JOB$$

$$JOB \triangleq job[\,\varnothing \parallel c_j \parallel \mathsf{out}\ srvi.\mathsf{inC}(v{:}\,\mathsf{data})clname.\langle \mathtt{file}(fname)\rangle^{\uparrow v}\,]$$

$$PWR \triangleq pwr[\,\{\mathsf{onMsg}^{\uparrow c_{srv}}\} \parallel c_{pwr} \parallel !\langle on\rangle^{\uparrow c_{srv}} \mid \mathsf{in}\ pwroff\,].$$

The purpose of *pwr* is simple. If it is present inside a server, it enables the service by continuously sending 'on' messages. However, if it is not present, the server is not able to listen to or respond to a request. Therefore, in order to shut a server down, the administrator should send a *POWER_OFF* message to that server:

$$POWER\_OFF(s) \triangleq pwroff[\,\varnothing \parallel c_{poff} \parallel \mathsf{in}\ cluster.\mathsf{in}\ s.\overline{\mathsf{in}}\,].$$

The *pwr* ambient would be locked inside *pwroff* after entering that ambient. Once inside *pwroff*, *pwr* is rendered inoperative. In fact, using algebraic properties we can show that

$$pwroff[\,\varnothing \parallel c_{poff} \parallel pwr[\,\{\mathsf{onMsg}^{\uparrow c_{srv}}\} \parallel c_{pwr} \parallel !\langle on\rangle^{\uparrow c_{srv}}\,]\,] \cong \mathbf{0},$$

and get rid of these *garbage* ambients.

Similarly, the administrator can restore the *pwr* ambient inside the server to 'power on' that server:

$$POWER\_ON\,(s) \triangleq pwron[\,\varnothing \parallel c_{pon} \parallel \mathsf{in}\ cluster.\mathsf{in}\ s.TURN\_ON\,]$$

$$TURN\_ON \triangleq pwr[\,\{\mathsf{onMsg}^{\uparrow c_{srv}}\} \parallel c_{pwr} \parallel \mathsf{out}\ pwron \mid !\langle on\rangle^{\uparrow c_{srv}} \mid \mathsf{in}\ pwroff\,].$$

Finally, we present a 'generic' client. The clients are generic in the sense that they do not need to know any of the port names in advance, since all of them are learnt on execution. The only requirement is that the client is well behaved and sends its own name in the request. A malicious client could send a different name. However, this can only cause a response to be lost or sent to the wrong client, which is unlikely since the malicious client needs to guess a correct client name.

$$CLIENT \triangleq client^{\rightleftharpoons}[\,\Gamma_{cl} \parallel c_{client} \parallel \mathsf{inC}(v_{clu}\colon(\mathsf{amb},\mathsf{filename}))cluster. \\ \langle client, afilename\rangle^{\uparrow v_{clu}}.\overline{\mathsf{inC}}(v_j : \mathsf{data}).(\mathsf{file})^{\downarrow v_j}.P \mid Q\,].$$

This is the basic structure of a client ambient. The port name can be changed without restrictions. The local view Γ_{cl} and the processes P and Q are arbitrary but for the restriction of not having conflicting types with the *cluster* and *job* ambients. The whole configuration looks like

$$SYSTEM \triangleq ADMIN \mid CLUSTER \mid CLIENTS.$$

The $ADMIN$ process could include processes like those in the power management and the $CLIENTS$ are also initially placed outside the cluster. As we have seen, they need to enter the cluster to get served.

4. Behavioural semantics

In order to study the behavioural semantics of **BACI**, we define an intuitive notion of barbed congruence (Milner and Sangiorgi 1992; Gordon and Cardelli 2003) based on the unlabelled reduction semantics given in Tables 2, 5, 6 and 7. We then introduce a labelled transition semantics inspired by Levi and Sangiorgi (2003), Merro and Hennessy (2006), Bugliesi *et al.* (2005) and Coppo *et al.* (2003), and prove that it coincides with unlabelled reduction. Finally, we define a notion of labelled bisimilarity and show that it is sound with respect to barbed congruence. The immediate benefit is that the co-inductive nature of bisimilarity can be exploited by putting its vast body of proof techniques to work in order to reason about barbed congruence. Since **BACI** has co-capabilities and allows parent-child communications, there are several reasonable choices of barbs, among which we have

$$P{\downarrow}^{1}_{(n)} \triangleq P \equiv (\boldsymbol{\nu}\tilde{m})(n[\,\Gamma_n \parallel c_n \parallel \overline{\mathsf{in}}.Q \mid R\,] \mid S) \tag{1}$$

$$P{\downarrow}^{2}_{(n)} \triangleq P \equiv (\boldsymbol{\nu}\tilde{m})(n[\,\Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(v : \tilde{\varphi}).Q \mid R\,] \mid S) \tag{2}$$

$$P{\downarrow}^{3}_{(c,c_n)} \triangleq P \equiv (\boldsymbol{\nu}\tilde{m})(n[\,\Gamma_n \parallel c_n \parallel (\tilde{x} : \tilde{\varphi})^{\uparrow c}.Q \mid R\,] \mid S) \tag{3}$$

$$P{\downarrow}^{4}_{(c,c_n)} \triangleq P \equiv (\boldsymbol{\nu}\tilde{m})(n[\,\Gamma_n \parallel c_n \parallel \langle \tilde{M}\rangle^{\uparrow c}.Q \mid R\,] \mid S) \tag{4}$$

provided that P is closed in all cases and $n \notin \tilde{m}$ in (1) and (2).

When possible we will use $P\downarrow^1_\Xi$ as shorthand for $P\downarrow^1_{(n)}$, which we call a 1-barb, and similarly for the other barbs. We write $P\Downarrow^i_\Xi$ (with $i \in [1\ldots4]$) if $P \twoheadrightarrow P'$ and $P'\downarrow^i_\Xi$, where $\twoheadrightarrow$ is the reflexive and transitive closure of $\longrightarrow$.

For each $i \in [1\ldots4]$, we define a reduction i-barbed congruence $\cong_i$ between processes, which takes into account the i-barbs. We require $\cong_i$ to be the largest equivalence relation that is preserved by typed contexts, and when restricted to closed processes it is reduction closed and i-barb preserving.

Definition 4.1. For $i \in [1\ldots4]$, $\dot{\approx}_i$*-barbed bisimilarity* is the largest symmetric relation, $\dot{\approx}_i$, such that whenever $P\dot{\approx}_i Q$:

— $P\downarrow^i_\Xi$ implies $Q\Downarrow^i_\Xi$;
— $P \longrightarrow P'$ implies $Q \twoheadrightarrow Q'$ for some $P'\dot{\approx}_i Q'$.

A typing environment Θ *extends* an environment Σ if Σ is a subset of Θ.

Definition 4.2 ($\{\Phi, d, \Delta\}/\{\Sigma, c, \Gamma\}$-context). Let Φ, Σ be typing environments, d, c be port names and Δ, Γ be local views. We say that a context $C[\cdot]$ is a $(\{\Phi, d, \Delta\}/\{\Sigma, c, \Gamma\})$-context if $\Phi \vdash_d C[\cdot] : \Delta$ is a valid type judgement when the hole $[\cdot]$ of $C[\cdot]$ is considered as a process and the following typing rule for $[\cdot]$ is added to the rules in Table 9:

$$(\{\Sigma, c, \Gamma\}\text{-HOLE})\quad \frac{\Theta \text{ extends } \Sigma}{\Theta \vdash_c [\cdot] : \Gamma}$$

Definition 4.3 (Barbed congruence). Let $i \in [1\ldots4]$. Two processes P, Q are *i-barbed congruent* ($P \cong_i Q$) if, for each typing environment Σ, port name c and local view Γ such that $\Sigma \vdash_c P : \Gamma$ and $\Sigma \vdash_c Q : \Gamma$, if $C[\cdot]$ is an arbitrary $\{\Phi, d, \Delta\}/\{\Sigma, c, \Gamma\}$-context, we have that $C[P]\dot{\approx}_i C[Q]$.

As expected, the four congruences coincide, so we can denote barbed congruence for **BACI** simply by $\cong$.

Theorem 4.4 (Independence from barbs). $\cong_i = \cong_j$ for all $i, j \in [1\ldots4]$.

Proof. The proof is by showing that the barbs imply each other by constructing, in each case, a context that relates two different barbs. The complete proof is deferred to Appendix B. □

4.1. *Labelled transition semantics*

This section presents a labelled transition semantics (LTS) and proves that it coincides with reduction. It is the first step towards a characterisation of reduction barbed congruence in terms of labelled bisimulation. The LTS is given in Tables 11–15. These tables define the labelled transition relation

$$P \xrightarrow{\xi} O$$

where

— P is a closed process,

Table 10. *Labels and outcomes*

$$
\begin{array}{llll}
\text{Labels} & \xi & ::= & (\tilde{M})^\eta \mid \langle - \rangle^\eta \mid \mathsf{get}(\tilde{M}, c, c') \mid \mathsf{put}(c, c') \mid \mathsf{pre\text{-}comm}(c) \\
& & & \mid \mathsf{in}\, n \mid \mathsf{out}\, n \mid \overline{\mathsf{in}} \mid \overline{\mathsf{out}} \mid \mathsf{in}(c : \rho, c')n \mid \mathsf{out}(c : \rho, c')n \\
& & & \mid [\![\Gamma \parallel c \parallel \overline{\mathsf{in}}\, n]\!] \mid \mathsf{pop}(c : \rho, c') \mid \mathsf{pre\text{-}exit}(c : \rho, c') \\
& & & \mid \mathsf{inC}(c : \tilde{\varphi})n \mid \mathsf{outC}(c : \tilde{\varphi})n \mid \overline{\mathsf{inC}}(c : \tilde{\varphi}) \mid \overline{\mathsf{outC}}(c_m : \tilde{\varphi}) \\
& & & \mid \mathsf{inC}(c : \tilde{\varphi}, c')n \mid \mathsf{outC}(c : \tilde{\varphi}, c')n \\
& & & \mid [\![\Gamma \parallel c \parallel \overline{\mathsf{inC}}(c' : \tilde{\varphi})n]\!] \mid \mathsf{popC}(c : \tilde{\varphi}, c') \mid \mathsf{pre\text{-}exitC}(c : \tilde{\varphi}, c') \\
& & & \mid \tau \\
\\
\text{Concretions} & K & ::= & (\boldsymbol{\nu} \tilde{p})\langle\!\langle P \rangle\!\rangle Q \mid (\boldsymbol{\nu} \tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P \\
\\
\text{Outcomes} & O & ::= & P \mid K
\end{array}
$$

Table 11. *Commitments: visible transitions (communication)*

(LTS SEND)

$$\frac{}{\langle \tilde{M} \rangle^\eta.P \xrightarrow{\langle - \rangle^\eta} \langle\!\langle \tilde{M} \rangle\!\rangle\, P}$$

(LTS RECV)

$$\frac{}{(\tilde{x} : \tilde{\varphi})^\eta.P \xrightarrow{(\tilde{M})^\eta} P\{\tilde{x} := \tilde{M}\}}$$

(LTS GET)

$$\frac{P \xrightarrow{(\tilde{M})^{\uparrow c_n}} P'}{m[\![\Gamma_m \parallel c_m \parallel P]\!] \xrightarrow{\mathsf{get}(\tilde{M}, c_n, c_m)} m[\![\Gamma_m \parallel c_m \parallel P']\!]}$$

(LTS PUT)

$$\frac{P \xrightarrow{\langle - \rangle^{\uparrow c_n}} (\boldsymbol{\nu} \tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P' \quad m \notin \tilde{p}}{m[\![\Gamma_m \parallel c_m \parallel P]\!] \xrightarrow{\mathsf{put}(c_n, c_m)} (\boldsymbol{\nu} \tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle m[\![\Gamma_m \parallel c_m \parallel P']\!]}$$

(LTS PRE-COMM-GET)

$$\frac{\begin{array}{c} P \xrightarrow{\langle - \rangle^{\downarrow c_m}} (\boldsymbol{\nu} \tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P' \\ Q \xrightarrow{\mathsf{get}(\tilde{M}, c_n, c_m)} Q' \quad \mathrm{fn}(Q) \cap \tilde{p} = \varnothing \end{array}}{P \mid Q \xrightarrow{\mathsf{pre\text{-}comm}(c_n)} (\boldsymbol{\nu} \tilde{p})(P' \mid Q')}$$

(LTS PRE-COMM-PUT)

$$\frac{\begin{array}{c} P \xrightarrow{\mathsf{put}(c_n, c_m)} (\boldsymbol{\nu} \tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P' \\ Q \xrightarrow{(\tilde{M})^{\downarrow c_m}} Q' \quad \mathrm{fn}(Q) \cap \tilde{p} = \varnothing \end{array}}{P \mid Q \xrightarrow{\mathsf{pre\text{-}comm}(c_n)} (\boldsymbol{\nu} \tilde{p})(P' \mid Q')}$$

— ξ is a label that encodes the interaction between P and the environment, and
— O is an 'outcome' resulting from that interaction.

Labels and outcomes are defined in Table 10.

An outcome may be a process P or a *concretion* $(\boldsymbol{\nu} \tilde{p})\langle\!\langle P \rangle\!\rangle Q$. Concretions are required to deal with transitions of components of the system that need to interact with the environment to be completed. Indeed, they prove convenient for formulating the silent transitions. In the concretion $(\boldsymbol{\nu} \tilde{p})\langle\!\langle P \rangle\!\rangle Q$, the process P is part of the system that interacts with the environment. For example, to complete an $\mathsf{in}\, n$ transition, the sibling ambient that hosts the entering one must be requested from the context. Similarly, in the concretion $(\boldsymbol{\nu} \tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle Q$, the message $\tilde{M}$ is the part of the system that interacts with the environment. This outcome is required only for the case of the transition for message output. In both

Table 12. *Commitments: visible transitions (mobility without port exchanges)*

(LTS CAP)
$$\frac{\zeta \in \{\mathsf{in}, \mathsf{out}\}}{\zeta\ n.P \xrightarrow{\zeta n} P}$$

(LTS COCAP)
$$\frac{\zeta \in \{\overline{\mathsf{in}}, \overline{\mathsf{out}}\}}{\zeta\ .P \xrightarrow{\zeta} P}$$

(LTS IN-OUT)
$$\frac{\zeta \in \{\mathsf{in}, \mathsf{out}\} \quad P \xrightarrow{\zeta n} P' \quad \Gamma_m(\uparrow c) = \rho}{m[\,\Gamma_m \parallel c_m \parallel P\,] \xrightarrow{\zeta(c\,:\,\rho, c_m)n} \langle\!\langle m[\,\Gamma_m \parallel c_m \parallel P'\,]\rangle\!\rangle\, \mathbf{0}}$$

(LTS COIN)
$$\frac{P \xrightarrow{\overline{\mathsf{in}}} P'}{n[\,\Gamma_n \parallel c_n \parallel P\,] \xrightarrow{[\Gamma_n \parallel c_n \parallel \overline{\mathsf{in}}\, n]} \langle\!\langle P'\rangle\!\rangle\, \mathbf{0}}$$

(LTS POP)
$$\frac{P \xrightarrow{\mathsf{out}(c\,:\,\rho, c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P'\rangle\!\rangle Q}{n[\,\Gamma_n \parallel c_n \parallel P\,] \xrightarrow{\mathsf{pop}(c\,:\,\rho, c_m)} (\boldsymbol{\nu}\tilde{p})(n[\,\Gamma_n \parallel c_n \parallel Q\,] \mid P')}$$

(LTS PRE-EXIT)
$$\frac{P \xrightarrow{\mathsf{pop}(c\,:\,\rho, c_m)} P' \quad Q \xrightarrow{\overline{\mathsf{out}}} Q'}{P \mid Q \xrightarrow{\mathsf{pre\text{-}exit}(c\,:\,\rho, c_m)} P' \mid Q'}$$

Table 13. *Commitments: visible transitions (mobility with port exchanges)*

(LTS CAPC)
$$\frac{\zeta \in \{\mathsf{inC}, \mathsf{outC}\}}{\zeta(v\!:\tilde{\varphi})n.P \xrightarrow{\zeta(c\,:\,\tilde{\varphi})n} P\{\uparrow c/\uparrow v\}}$$

(LTS IN-OUTC)
$$\frac{\zeta \in \{\mathsf{inC}, \mathsf{outC}\} \quad P \xrightarrow{\zeta(c\,:\,\tilde{\varphi})n} P' \qquad \Gamma'_m = \Gamma_m \oplus \tilde{\varphi}^{\uparrow c} \text{ is defined}}{m[\,\Gamma_m \parallel c_m \parallel P\,] \xrightarrow{\zeta(c\,:\,\tilde{\varphi}, c_m)n} \langle\!\langle m[\,\Gamma'_m \parallel c_m \parallel P'\,]\rangle\!\rangle\, \mathbf{0}}$$

(LTS COCAPC)
$$\frac{\overline{\zeta} \in \{\overline{\mathsf{inC}}, \overline{\mathsf{outC}}\}}{\overline{\zeta}(v : \tilde{\varphi}).P \xrightarrow{\overline{\zeta}(c_m\,:\,\tilde{\varphi})} P\{\downarrow c_m/\downarrow v\}}$$

(LTS COINC)
$$\frac{P \xrightarrow{\overline{\mathsf{inC}}(c_m\,:\,\tilde{\varphi})} P' \quad \Gamma_n \oplus \tilde{\varphi}^{\downarrow c_m} \text{ is defined}}{n[\,\Gamma_n \parallel c_n \parallel P\,] \xrightarrow{[\Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(c_m\,:\,\tilde{\varphi})n]} \langle\!\langle P'\rangle\!\rangle\, \mathbf{0}}$$

(LTS PRE-EXITC)
$$\frac{P \xrightarrow{\mathsf{popC}(c\,:\,\tilde{\varphi}, c_m)} P' \qquad Q \xrightarrow{\overline{\mathsf{outC}}(c_m\,:\,\tilde{\varphi})} Q'}{P \mid Q \xrightarrow{\mathsf{pre\text{-}exitC}(c\,:\,\tilde{\varphi}, c_m)} P' \mid Q'}$$

(LTS POPC)
$$\frac{P \xrightarrow{\mathsf{outC}(c\,:\,\tilde{\varphi}, c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P'\rangle\!\rangle Q}{n[\,\Gamma_n \parallel c_n \parallel P\,] \xrightarrow{\mathsf{popC}(c\,:\,\tilde{\varphi}, c_m)} (\boldsymbol{\nu}\tilde{p})(n[\,\Gamma_n \parallel c_n \parallel Q\,] \mid P')}$$

cases, Q represents the remaining part of the process that is not affected by the transition. The transitions are inspired by those of Bugliesi *et al.* (2005) and Coppo *et al.* (2003).

The τ transitions for message exchanges are (LTS τ-LOCALCOMM) for local exchange and (LTS τ-COMM-$\downarrow$) for non-local exchange. Rule (LTS τ-COMM-$\downarrow$) uses the label $\mathsf{pre\text{-}comm}$ generated by rules (LTS PRE-COMM-GET) and (LTS PRE-COMM-PUT). In (LTS PRE-COMM-GET), the directed input action towards the child ambient must be met by a corresponding output action from the child. In (LTS PRE-COMM-PUT), the input and output roles are exchanged.

The τ transitions for mobility are (LTS τ-ENTER), (LTS τ-ENTERC), (LTS τ-EXIT) and (LTS τ-EXITC). Since these are similar in spirit, we shall confine our discussion to (LTS τ-ENTERC). Rule (LTS τ-ENTERC) is in charge of synchronising two actions, namely the request by an ambient to enter a host ambient with the action witnessing the approval (by means of an appropriate co-action) on the part of the host ambient. Therefore, the label

Table 14. *Commitments: τ transitions*

(LTS τ-LocalComm)

$$\frac{\begin{array}{c} P \xrightarrow{\langle - \rangle^\star} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P' \\ Q \xrightarrow{(\tilde{M})^\star} Q' \quad \mathrm{fn}(Q) \cap \tilde{p} = \varnothing \end{array}}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p})(P' \mid Q')}$$

(LTS τ-Comm-$\downarrow$)

$$\frac{P \xrightarrow{\mathsf{pre\text{-}comm}(c_n)} P'}{n[\![\, \Gamma_n \parallel c_n \parallel P \,]\!] \xrightarrow{\tau} n[\![\, \Gamma_n \parallel c_n \parallel P' \,]\!]}$$

(LTS τ-Enter)

$$\frac{\begin{array}{c} P \xrightarrow{\mathsf{in}(c_n \,:\, \rho, c_m) n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m \rangle\!\rangle P' \\ Q \xrightarrow{[\![\, \Gamma_n \parallel c_n \parallel \overline{\mathsf{in}}\ n \,]\!]} (\boldsymbol{\nu}\tilde{q})\langle\!\langle Q_n \rangle\!\rangle Q' \\ \mathrm{fn}(P) \cap \tilde{q} = \mathrm{fn}(Q) \cap \tilde{p} = \tilde{p} \cap \tilde{q} = \varnothing \\ \rho \preceq \Gamma_n(\downarrow c_m) \end{array}}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p}, \tilde{q})(n[\![\, \Gamma_n \parallel c_n \parallel Q_n \mid P_m \,]\!] \mid P' \mid Q')}$$

(LTS τ-Exit)

$$\frac{\begin{array}{c} P \xrightarrow{\mathsf{pre\text{-}exit}(c_p \,:\, \rho, c_m)} P' \\ \rho \preceq \Gamma_p(\downarrow c_m) \end{array}}{p[\![\, \Gamma_p \parallel c_p \parallel P \,]\!] \xrightarrow{\tau} p[\![\, \Gamma_p \parallel c_p \parallel P' \,]\!]}$$

(LTS τ-EnterC)

$$\frac{\begin{array}{c} P \xrightarrow{\mathsf{inC}(c_n \,:\, \tilde{\varphi}, c_m) n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m \rangle\!\rangle P' \\ Q \xrightarrow{[\![\, \Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(c_m \,:\, \tilde{\varphi}) n \,]\!]} (\boldsymbol{\nu}\tilde{q})\langle\!\langle Q_n \rangle\!\rangle Q' \\ \mathrm{fn}(P) \cap \tilde{q} = \mathrm{fn}(Q) \cap \tilde{p} = \tilde{p} \cap \tilde{q} = \varnothing \end{array}}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p}, \tilde{q})(n[\![\, \Gamma_n \oplus \tilde{\varphi}^{\downarrow c_m} \parallel c_n \parallel Q_n \mid P_m \,]\!] \mid P' \mid Q')}$$

(LTS τ-ExitC)

$$\frac{\begin{array}{c} P \xrightarrow{\mathsf{pre\text{-}exitC}(c_p \,:\, \tilde{\varphi}, c_m)} P' \\ \Gamma'_p = \Gamma_p \oplus \tilde{\varphi}^{\downarrow c_m} \text{ is defined} \end{array}}{p[\![\, \Gamma_p \parallel c_p \parallel P \,]\!] \xrightarrow{\tau} p[\![\, \Gamma'_p \parallel c_p \parallel P' \,]\!]}$$

Table 15. *Commitments: structural transitions*

(LTS Par)

$$\frac{P \xrightarrow{\zeta} O}{P \mid Q \xrightarrow{\zeta} O \mid Q}$$

(LTS Path)

$$\frac{C.(D.P) \xrightarrow{\zeta} P'}{(C.D).P \xrightarrow{\zeta} P'}$$

(LTS Res)

$$\frac{P \xrightarrow{\zeta} O \quad \tilde{n} \cap \mathrm{fn}(\zeta) = \varnothing}{(\boldsymbol{\nu}\tilde{n})P \xrightarrow{\zeta} (\boldsymbol{\nu}\tilde{n})O}$$

(LTS Amb)

$$\frac{P \xrightarrow{\tau} P'}{n[\![\, \Gamma_n \parallel c_n \parallel P \,]\!] \xrightarrow{\tau} n[\![\, \Gamma_n \parallel c_n \parallel P' \,]\!]}$$

(LTS Repl)

$$\frac{\pi.P \xrightarrow{\zeta} O}{!\pi.P \xrightarrow{\zeta} O \mid !\pi.P}$$

of the first action is $\mathsf{inC}(c_n : \tilde{\varphi}, c_m)n$, while that of the second is $[\![\, \Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(c_m : \tilde{\varphi})n \,]\!]$. The former records the communication ports of both the moving and destination ambients, their topic of conversation and the name of the destination ambient. The latter gives the same information plus the local view Γ_n of the destination ambient to which we add $\tilde{\varphi}^{\downarrow c_m}$. We know that $\Gamma_n \oplus \tilde{\varphi}^{\downarrow c_m}$ is defined by rule (LTS coInC). The process that actually moves is represented by P_m in the concretion resulting from the first action, while Q_n represents the process that runs alongside the visiting ambient. The processes P' and Q' are the sub-components of P and Q that do not participate in the movement. A third premise of the rule guarantees that free ambient names are not erroneously captured by the name restrictions on the conclusion of the rule.

Note that the transition (LTS REPL) is sound only because we do not allow full replication. For example, the expression $!n[\varnothing \parallel c_n \parallel \mathsf{in}\ n \mid \overline{\mathsf{in}}]$ is not a pre-process according to the syntax of Table 1.

By comparing the notion of observability (*cf.* the definition of barbs) with rule (LTS COIN), one can easily see that a name is observable if and only if the action $[\Gamma \parallel c \parallel \overline{\mathsf{in}}\, n]$ can be performed. In particular, we have the following lemma.

Lemma 4.5. $P \downarrow^1_{(n)}$ if and only if $P \xrightarrow{[\Gamma_n \parallel c_n \parallel \overline{\mathsf{in}}\ n]} O$ for some Γ_n, c_n and O.

Proof. The proof of this lemma and its supporting lemmas are in Appendix C. □

A similar observation applies to rules (LTS COINC), (LTS GET), (LTS PUT) and the observability of co-in with port exchanges or of pairs of port names (*cf.* barbs (2), (3) and (4) at the start of Section 4). Thanks to Theorem 4.4, we need only consider one notion of barb.

As expected, unlabelled reduction and labelled reduction coincide.

Theorem 4.6. Let P be closed.

1 If $P \xrightarrow{\tau} P'$, then $P \longrightarrow P'$.

2 If $P \longrightarrow P'$, then $P \xrightarrow{\tau} Q$ and $Q \equiv P'$ for some Q.

Proof. The proof of the first statement is by induction on the transition rules. For the second statement, the proof is by induction on the reduction rules. See Appendix D for the proof of this theorem. □

4.2. *Full bisimilarity and its soundness*

This section defines a notion of labelled bisimilarity and shows that it is sound with respect to reduction barbed congruence. Labelled bisimilarity requires checking when two processes produce equal observable actions. The problem is that the current definition of labelled reduction may produce a concretion instead of a process. This situation is remedied by introducing *higher-order (HO) transitions* (Merro and Hennessy 2006) for those labelled transitions of Tables 11–13 that produce a concretion as an outcome.

The HO-transitions are given in Tables 16–18. In these transitions we use richer labels obtained by adding to some of the previous labels ξ a new component (prefixed by $\diamond$), which can have one of the following three shapes:

— P;
— $[\Gamma \parallel c \parallel P]$;
— $n[\Gamma \parallel c \parallel P] \mid Q$.

This component describes the minimum contribution of the context necessary to fire the transition. For example, in rule (HO OUT) the context must provide the three components (local view, port and process) of the ambient n from which the process P_m exits and in which the process P' remains.

Table 16. *Commitments: higher-order transitions (communication)*

(HO Send $\star$-$\Downarrow$)

$$\frac{P \xrightarrow{\langle - \rangle^{\eta}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P' \quad \begin{array}{c} \eta \in \{\star, \downarrow c'\} \\ \mathrm{fn}(Q) \cap \tilde{p} = \varnothing \\ \mathrm{fv}(Q) \subseteq \tilde{x} \end{array}}{P \xrightarrow{\langle - \rangle^{\eta} \diamond Q} (\boldsymbol{\nu}\tilde{p})(P' \mid Q\{\tilde{x} := \tilde{M}\})}$$

(HO Send$^{\uparrow}$)

$$\frac{P \xrightarrow{\langle - \rangle^{\uparrow c}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P' \quad \begin{array}{c} (\mathrm{fn}(Q) \cup \mathrm{fn}(R) \cup \{m\}) \cap \tilde{p} = \varnothing \\ \mathrm{fv}(Q) \subseteq \tilde{x} \\ \mathrm{fv}(R) \cap \tilde{x} = \varnothing \end{array}}{P \xrightarrow{\langle - \rangle^{\uparrow c} \diamond m[\Gamma_m \parallel c_m \parallel R] \mid Q} (\boldsymbol{\nu}\tilde{p})(m[\Gamma_m \parallel c_m \parallel P' \mid R] \mid Q\{\tilde{x} := \tilde{M}\})}$$

(HO Put)

$$\frac{P \xrightarrow{\mathsf{put}(c_n, c_m)} (\boldsymbol{\nu}\tilde{q})\langle\!\langle \tilde{M} \rangle\!\rangle P' \quad \begin{array}{c} \mathrm{fn}(Q) \cap \tilde{p} = \varnothing \\ \mathrm{fv}(Q) \subseteq \tilde{x} \end{array}}{P \xrightarrow{\mathsf{put}(c_n, c_m) \diamond Q} (\boldsymbol{\nu}\tilde{q})(P' \mid Q\{\tilde{x} := \tilde{M}\})}$$

Table 17. *Commitments: higher-order transitions (mobility without port exchanges)*

(HO In)

$$\frac{P \xrightarrow{\mathsf{in}(c_n : \rho, c_m) n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m \rangle\!\rangle P' \quad \mathrm{fn}(Q) \cap \tilde{p} = \varnothing}{P \xrightarrow{\mathsf{in}(c_n : \rho, c_m) n \diamond [\Gamma_n \parallel c_n \parallel Q]} (\boldsymbol{\nu}\tilde{p})(n[\Gamma_n \parallel c_n \parallel P_m \mid Q] \mid P')}$$

(HO Out)

$$\frac{P \xrightarrow{\mathsf{out}(c_q : \rho, c_m) n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m \rangle\!\rangle P' \quad \mathrm{fn}(Q) \cap \tilde{p} = \varnothing}{P \xrightarrow{\mathsf{out}(c_q : \rho, c_m) n \diamond [\Gamma_n \parallel c_n \parallel Q]} (\boldsymbol{\nu}\tilde{p})(P_m \mid n[\Gamma_n \parallel c_n \parallel P' \mid Q])}$$

(HO Co-In)

$$\frac{P \xrightarrow{[\Gamma_n \parallel c_n \parallel \overline{\mathsf{in}}\, n]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \mathrm{fn}(Q_m) \cap \tilde{p} = \varnothing}{P \xrightarrow{[\Gamma_n \parallel c_n \parallel \overline{\mathsf{in}}\, n] \diamond Q_m} (\boldsymbol{\nu}\tilde{p})(n[\Gamma_n \parallel c_n \parallel P_1 \mid Q_m] \mid P_2)}$$

Since we always consider well-formed processes, in the higher-order labels we only allow processes, port names and local views to ensure that the processes obtained by the transition are well formed.

For HO transitions we get the following version of Lemma 4.5.

Lemma 4.7. $P \downarrow^{1}_{(n)}$ if and only if $P \xrightarrow{[\Gamma \parallel c \parallel \overline{\mathsf{in}}\ n] \diamond Q} P'$ for some Γ, c, Q and P'.

Proof. The proof is straightforward by inspecting the higher-order transitions and Lemma 4.5. $\square$

Table 18. *Commitments: higher-order transitions (mobility with port exchanges)*

(HO INC)

$$\frac{P \xrightarrow{\mathsf{inC}(c_n : \tilde{\varphi}, c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m \rangle\!\rangle P' \qquad \mathrm{fn}(Q) \cap \tilde{p} = \varnothing}{P \xrightarrow{\mathsf{inC}(c_n : \tilde{\varphi}, c_m)n \diamond [\Gamma_n \parallel c_n \parallel Q]} (\boldsymbol{\nu}\tilde{p})(n[\Gamma_n \parallel c_n \parallel P_m \mid Q] \mid P')}$$

(HO OUTC)

$$\frac{P \xrightarrow{\mathsf{outC}(c_q : \tilde{\varphi}, c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m \rangle\!\rangle P' \qquad \mathrm{fn}(Q) \cap \tilde{p} = \varnothing}{P \xrightarrow{\mathsf{outC}(c_q : \tilde{\varphi}, c_m)n \diamond [\Gamma_n \parallel c_n \parallel Q]} (\boldsymbol{\nu}\tilde{p})(P_m \mid n[\Gamma_n \parallel c_n \parallel P' \mid Q])}$$

(HO CO-INC)

$$\frac{P \xrightarrow{[\Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(c_m : \tilde{\varphi})n]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \qquad \mathrm{fn}(Q_m) \cap \tilde{p} = \varnothing}{P \xrightarrow{[\Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(c_m : \tilde{\varphi})n] \diamond Q_m} (\boldsymbol{\nu}\tilde{p})(n[\Gamma_n \parallel c_n \parallel P_1 \mid Q_m] \mid P_2)}$$

We use Λ to denote the set of labels that includes both the first-order labels defined in Table 10, and the HO labels of Tables 16–18. In the following, we let λ range over Λ and $\Longrightarrow$ denote the reflexive and transitive closure of $\xrightarrow{\tau}$. We adopt the following notation:

1 $\overset{\lambda}{\Longrightarrow}$ denotes $\Longrightarrow\xrightarrow{\lambda}\Longrightarrow$.
2 $\overset{\hat{\lambda}}{\Longrightarrow}$ denotes $\Longrightarrow$ if $\lambda = \tau$, and $\overset{\lambda}{\Longrightarrow}$ otherwise.

As a final step towards defining bisimilarity, we adapt the notion of a typed relation to our setting. A *typed relation* $\mathscr{R}$ is a set of pairs of the form $(P\,; Q)$ where P and Q are two closed processes such that $\varnothing \vdash_c P : \Gamma$ implies $\varnothing \vdash_c Q : \Gamma$ for any c, Γ and *vice versa*. We use the notation $P\mathscr{R}Q$ when $(P\,; Q) \in \mathscr{R}$.

Definition 4.8 (Bisimilarity).

1 A symmetric typed relation $\mathscr{R}$ over closed processes is a *bisimulation* if $P\mathscr{R}Q$ and $P \xrightarrow{\lambda} P'$ imply that there exists Q' such that:

— $Q \overset{\hat{\lambda}}{\Longrightarrow} Q'$; and

— $P'\mathscr{R}Q'$.

2 We say that two closed processes P and Q are *bisimilar*, written $P \approx Q$, if $P\mathscr{R}Q$ for some bisimulation $\mathscr{R}$.

Since variables in processes can only stand for ambient names, port names or (co)-capabilities, we can immediately define closing substitutions that respect types.

The definition of bisimulation is extended to arbitrary processes as usual.

Definition 4.9 (Full bisimilarity). Two processes P and Q are *fully bisimilar*, written $P \approx_c Q$, if $P\mathsf{s} \approx Q\mathsf{s}$ for every closing substitution s that respects types.

Following the proof scheme of Bugliesi *et al.* (2005) and Merro and Hennessy (2006), we can show that full bisimilarity is preserved by context.

Theorem 4.10. Full bisimilarity is a congruence.

Proof. See Appendix E. □

Finally, we prove that $\approx_c$ is contained in $\cong$, as desired.

Theorem 4.11 (Soundness of full bisimilarity). If $P \approx_c Q$, then $P \cong Q$.

Proof. See Appendix F. □

We conjecture the incompleteness of $\approx_c$ for the same reason as the authors of Bugliesi *et al.* (2005) conjecture the incompleteness of the full bisimilarity arising from a similar LTS for NBA, namely the difficulty of finding a context that discriminates the label $\langle \tilde{M} \rangle^{\uparrow c}$. We also conjecture that an LTS for **BACI** inducing a complete full bisimilarity could be developed by following the approach of (Bugliesi *et al.* 2005).

4.3. *Algebraic laws*

This section presents some algebraic laws that give a better account of the semantics of processes in **BACI**. These and other laws can be proved by means of the labelled bisimilarity developed in the previous subsections.

The laws holding in **BACI** that deal with mobility are very similar to those true for the NBA calculus (Bugliesi *et al.* 2005), so we will not discuss them further here.

Instead, **BACI**'s refined treatment of communication using port names allows us to get quite interesting laws concerning input–output. For example, an ambient that is only willing to communicate with its father, but using a 'wrong' port name, is dead, that is, we have the following *garbage collection laws*:

$$n[\![\,\Gamma_n \parallel c_n \parallel m[\![\,\Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi})^{\uparrow c}.P\,]\!] \mid Q\,]\!] \quad \cong \quad n[\![\,\Gamma_n \parallel c_n \parallel Q\,]\!]$$

$$n[\![\,\Gamma_n \parallel c_n \parallel m[\![\,\Gamma_m \parallel c_m \parallel \langle \tilde{M} \rangle^{\uparrow c}.P\,]\!] \mid Q\,]\!] \quad \cong \quad n[\![\,\Gamma_n \parallel c_n \parallel Q\,]\!]\,.$$

In NBA, a parent–child communication can be forced only if the processes involved in the communication are the only active processes inside both ambients. In **BACI**, however, there can be other active processes provided they do not know the port name of the communication partner and some ambient names do not occur in some processes or they are restricted. The conditions on port names avoid interfering communications, and the conditions on ambient names avoid interfering movements. More precisely, we have the following.

If c_m does not occur in S, and hence no process in S can communicate with a process inside the ambient m,

$$(\boldsymbol{\nu} n)(n[\![\,\Gamma_n \parallel c_n \parallel m[\![\,\Gamma_m \parallel c_m \parallel \langle \tilde{M} \rangle^{\uparrow c_n}.P\,]\!] \mid (\tilde{x} : \tilde{\varphi})^{\downarrow c_m}.Q \mid S\,]\!])$$

$$\cong$$

$$(\boldsymbol{\nu} n)(n[\![\,\Gamma_n \parallel c_n \parallel m[\![\,\Gamma_m \parallel c_m \parallel P\,]\!] \mid Q\{\tilde{x} := \tilde{M}\} \mid S\,]\!])$$

and

$$(\boldsymbol{\nu} n)(n[\![\, \Gamma_n \parallel c_n \parallel m[\![\, \Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi})^{\uparrow c_n}.P \,]\!] \mid \langle \tilde{M} \rangle^{\downarrow c_m}.Q \mid S \,]\!])$$
$$\cong$$
$$(\boldsymbol{\nu} n)(n[\![\, \Gamma_n \parallel c_n \parallel m[\![\, \Gamma_m \parallel c_m \parallel P\{\tilde{x} := \tilde{M}\} \,]\!] \mid Q \mid S \,]\!]).$$

If c_n and n do not occur in R,

$$n[\![\, \Gamma_n \parallel c_n \parallel (\boldsymbol{\nu} m)(m[\![\, \Gamma_m \parallel c_m \parallel \langle \tilde{M} \rangle^{\uparrow c_n}.P \mid R \,]\!]) \mid (\tilde{x} : \tilde{\varphi})^{\downarrow c_m}.Q \,]\!]$$
$$\cong$$
$$(\boldsymbol{\nu} m)(n[\![\, \Gamma_n \parallel c_n \parallel m[\![\, \Gamma_m \parallel c_m \parallel P \mid R \,]\!] \mid Q\{\tilde{x} := \tilde{M}\} \,]\!])$$

and

$$n[\![\, \Gamma_n \parallel c_n \parallel (\boldsymbol{\nu} m)(m[\![\, \Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi})^{\uparrow c_n}.P \mid R \,]\!]) \mid \langle \tilde{M} \rangle^{\downarrow c_m}.Q \,]\!]$$
$$\cong$$
$$n[\![\, \Gamma_n \parallel c_n \parallel (\boldsymbol{\nu} m)(m[\![\, \Gamma_m \parallel c_m \parallel P\{\tilde{x} := \tilde{M}\} \mid R \,]\!] \mid Q \,]\!]).$$

If R contains a process that could communicate with a process inside the ambient n, this process would contain c_n. If R contains a process that could take ambient m out of ambient n, this process would contain n.

Note that in the following group of equivalences, R cannot contain m, since if it did, an ambient inside R could exit m and communicate the port name c_m to the process S.

If c_m does not occur in S, and c_n, n, m do not occur in R, then

$$(\boldsymbol{\nu} n)(n[\![\, \Gamma_n \parallel c_n \parallel (\boldsymbol{\nu} m)(m[\![\, \Gamma_m \parallel c_m \parallel \langle \tilde{M} \rangle^{\uparrow c_n}.P \mid R \,]\!]) \mid (\tilde{x} : \tilde{\varphi})^{\downarrow c_m}.Q \mid S \,]\!])$$
$$\cong$$
$$(\boldsymbol{\nu} n)(\boldsymbol{\nu} m)(n[\![\, \Gamma_n \parallel c_n \parallel m[\![\, \Gamma_m \parallel c_m \parallel P \mid R \,]\!] \mid Q\{\tilde{x} := \tilde{M}\} \mid S \,]\!])$$

and

$$(\boldsymbol{\nu} n)(n[\![\, \Gamma_n \parallel c_n \parallel (\boldsymbol{\nu} m)(m[\![\, \Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi})^{\uparrow c_n}.P \mid R \,]\!]) \mid \langle \tilde{M} \rangle^{\downarrow c_m}.Q \mid S \,]\!])$$
$$\cong$$
$$(\boldsymbol{\nu} n)(n[\![\, \Gamma_n \parallel c_n \parallel (\boldsymbol{\nu} m)(m[\![\, \Gamma_m \parallel c_m \parallel P\{\tilde{x} := \tilde{M}\} \mid R \,]\!]) \mid Q \mid S \,]\!]).$$

Each law can be proved by exhibiting the bisimulation $\{(LHS, RHS)\} \cup \mathscr{I}$, where LHS and RHS denote the left- and right-hand sides of the equation and $\mathscr{I}$ is the identity.

5. Conclusions

We have presented a typed calculus of mobile ambients that features both local and dynamic typing. Each ambient comes equipped with a local communication interface consisting of a communication port and a local view indicating the type of information that may be exchanged over parent and child ports. In addition to the usual communication within an ambient, messages may be exchanged across ambient boundaries. The type system guarantees that in this case the types of the local ports of the sending and receiving ambients agree. Since communication interfaces are local and ambients may migrate, ambients must be able to increase their local knowledge of their surroundings. Therefore, the mobility rules allow an ambient to learn the communication type of the local port of the ambient that it enters. Appropriate run-time checks are required so that the entering and host ambients agree on a topic of conversation. Among the novel aspects of **BACI** are:

- *Communicating ports*. In contrast with previous ambient calculi, **BACI** uses names for mobility and ports for communication.
- *Named communication with parents*. In previous calculi communication with a parent was decided by the location of an ambient, but in **BACI** the communication with a parent is indexed by the parent's port, in a way similar to that in which communication with a child is usually indexed. This new named communication allows an ambient to communicate with different parents at different types.
- *Finer control of non-determinism*. The division between names and ports introduces the ability to have non-determinism for mobility and determinism for communication and *vice versa*, while in previous calculi this was not possible.
- *Local typing*. Having different TOCs with different parents allows us to control which parent can exchange information, while in previous calculi the type of a communication with the parent remained fixed.

Although communication control is local, this is not the case for mobility. Mobility is currently unrestricted, which poses the question of whether one might also include, in the local knowledge of an ambient, some indication of whether the ambient is allowed to move. Among the possible enhancements for **BACI** are the inclusion of access control mechanisms to regulate mobility and the extension of **BACI**'s type system with correspondence assertions to describe more complex protocols for port use.

Appendix A. Subject Reduction (Theorem 2.1)

Table A gives the full definition of structural congruence. We start with some preliminary lemmas.

Lemma A.1 (Subject congruence).
(1) If $\Sigma \vdash_c P : \Gamma$ and $P \equiv Q$, then $\Sigma \vdash_c Q : \Gamma$.
(2) If $\Sigma \vdash_c Q : \Gamma$ and $P \equiv Q$, then $\Sigma \vdash_c P : \Gamma$.

Proof. The proof is by simultaneous induction on the derivations of $\Sigma \vdash_c P : \Gamma$ and $\Sigma \vdash_c Q : \Gamma$. Stating the lemma in its two parts allows us to deal easily with the case of rule (Struct Symm). The other inductive cases are immediate by induction, so we will just focus on the interesting base cases.

(Struct Rep Par)

(1) The derivation of $\Sigma \vdash_c P : \Gamma$ ends in

$$\frac{\Sigma \vdash_c R : \Gamma}{\Sigma \vdash_c !R : \Gamma}\ (\textsc{Rep})$$

We may derive

$$\dfrac{\Sigma \vdash_c R : \Gamma \qquad \dfrac{\Sigma \vdash_c R : \Gamma}{\Sigma \vdash_c !R : \Gamma}\ (\textsc{Rep})}{\Sigma \vdash_c R \mid !R : \Gamma}\ (\textsc{Comp})$$

Table A. *Structural congruence (full definition)*

$P \equiv P$	(Struct Refl)
$P \equiv Q \implies Q \equiv P$	(Struct Symm)
$P \equiv Q, Q \equiv R \implies P \equiv R$	(Struct Trans)
$P \equiv Q \implies (\boldsymbol{\nu} n)P \equiv (\boldsymbol{\nu} n)Q$	(Struct Res)
$P \equiv Q \implies P \mid R \equiv Q \mid R$	(Struct Par)
$\pi.P \equiv \pi.Q \implies !\pi.P \equiv !\pi.Q$	(Struct Rep)
$P \equiv Q \implies n[\![\, \Gamma_n \parallel c_n \parallel P \,]\!] \equiv n[\![\, \Gamma_n \parallel c_n \parallel Q \,]\!]$	(Struct Amb)
$P \equiv Q \implies \pi.P \equiv \pi.Q$	(Struct Prefix)
$P \mid Q \equiv Q \mid P$	(Struct Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)
$!\pi.P \equiv \pi.P \mid !\pi.P$	(Struct Rep Par)
$(\boldsymbol{\nu} n)(\boldsymbol{\nu} m)P \equiv (\boldsymbol{\nu} m)(\boldsymbol{\nu} n)P$	(Struct Res Res)
$(\boldsymbol{\nu} n)(P \mid Q) \equiv P \mid (\boldsymbol{\nu} n)Q$, if $n \notin \mathrm{fn}(P)$	(Struct Res Par)
$(\boldsymbol{\nu} n)m[\![\, \Gamma_m \parallel c_m \parallel P \,]\!] \equiv m[\![\, \Gamma_m \parallel c_m \parallel (\boldsymbol{\nu} n)P \,]\!]$, if $n \neq m$	(Struct Res Amb)
$P \mid \mathbf{0} \equiv P$	(Struct Zero Par)
$(\boldsymbol{\nu} n)\mathbf{0} \equiv \mathbf{0}$	(Struct Zero Res)
$(C.D).P \equiv C.D.P$	(Struct .)

(2) Note that a derivation of $\Sigma \vdash_c R \mid !R : \Gamma$ includes, as a subderivation, a derivation of $\Sigma \vdash_c R : \Gamma$.

(Struct Res Res)

(1) The derivation of $\Sigma \vdash_c P : \Gamma$ ends in

$$\dfrac{\dfrac{\Sigma \vdash_c R : \Gamma}{\Sigma \vdash_c (\boldsymbol{\nu} m)R : \Gamma}\ (\textsc{Res})}{\Sigma \vdash_c (\boldsymbol{\nu} n)(\boldsymbol{\nu} m)R : \Gamma}\ (\textsc{Res})$$

Thus we derive

$$\dfrac{\dfrac{\Sigma \vdash_c R : \Gamma}{\Sigma \vdash_c (\boldsymbol{\nu} n)R : \Gamma}\ (\textsc{Res})}{\Sigma \vdash_c (\boldsymbol{\nu} m)(\boldsymbol{\nu} n)R : \Gamma}\ (\textsc{Res})$$

(Struct Res Par)

(1) The derivation of $\Sigma \vdash_c P : \Gamma$ ends in

$$\dfrac{\dfrac{\Sigma \vdash_c R_1 : \Gamma \quad \Sigma \vdash_c R_2 : \Gamma}{\Sigma \vdash_c R_1 \mid R_2 : \Gamma}\ (\textsc{Comp})}{\Sigma \vdash_c (\boldsymbol{\nu} n)(R_1 \mid R_2) : \Gamma}\ (\textsc{Res})$$

We derive

$$\dfrac{\Sigma \vdash_c R_1 : \Gamma \quad \dfrac{\Sigma \vdash_c R_2 : \Gamma}{\Sigma \vdash_c (\boldsymbol{\nu} n)R_2 : \Gamma}\,(\textsc{Res})}{\Sigma \vdash_c R_1 \mid (\boldsymbol{\nu} n)R_2 : \Gamma}\,(\textsc{Comp})$$

(Struct Res Amb)

(1) The derivation of $\Sigma \vdash_c P : \Gamma$ ends in

$$\dfrac{\dfrac{\Sigma \vdash_{c_m} R : \Gamma_m \quad \Sigma \vdash m : \mathsf{amb} \quad \Gamma_m(\uparrow c) \preceq \Gamma(\downarrow c_m) \quad \Gamma_m \text{ is closed}}{\Sigma \vdash_c m[\![\Gamma_m \parallel c_m \parallel R]\!] : \Gamma}\,(\textsc{Amb})}{\Sigma \vdash_c (\boldsymbol{\nu} n)m[\![\Gamma_m \parallel c_m \parallel R]\!] : \Gamma}\,(\textsc{Res})$$

We derive

$$\dfrac{\dfrac{\Sigma \vdash_{c_m} R : \Gamma_m}{\Sigma \vdash_{c_m} (\boldsymbol{\nu} n)R : \Gamma_m}\,(\textsc{Res}) \quad \Sigma \vdash m : \mathsf{amb} \quad \Gamma_m(\uparrow c) \preceq \Gamma(\downarrow c_m) \quad \Gamma_m \text{ is closed}}{\Sigma \vdash_c m[\![\Gamma_m \parallel c_m \parallel (\boldsymbol{\nu} n)R]\!] : \Gamma}\,(\textsc{Amb})$$

(Struct Zero Par)

(1) The derivation of $\Sigma \vdash_c P : \Gamma$ ends in

$$\dfrac{\Sigma \vdash_c Q : \Gamma \quad \Sigma \vdash_c \mathbf{0} : \Gamma}{\Sigma \vdash_c Q \mid \mathbf{0} : \Gamma}\,(\textsc{Comp})$$

and we already have as hypothesis a derivation of $\Sigma \vdash_c Q : \Gamma$.

(2) Suppose we have a derivation of $\Sigma \vdash_c Q : \Gamma$. Thus we may derive $\Sigma \vdash_c P : \Gamma$ as follows:

$$\dfrac{\Sigma \vdash_c Q : \Gamma \quad \dfrac{}{\Sigma \vdash_c \mathbf{0} : \Gamma}\,(\textsc{Inact})}{\Sigma \vdash_c Q \mid \mathbf{0} : \Gamma}\,(\textsc{Comp})$$

(Struct Zero Res)

(1) The derivation of $\Sigma \vdash_c P : \Gamma$ ends in

$$\dfrac{\dfrac{}{\Sigma \vdash_c \mathbf{0} : \Gamma}\,(\textsc{Inact})}{\Sigma \vdash_c (\boldsymbol{\nu} n)\mathbf{0} : \Gamma}\,(\textsc{Res})$$

and we may conclude by taking the subderivation of $\Sigma \vdash_c \mathbf{0} : \Gamma$. □

We now generalise the definition of location substitution on processes given in Table 3 in two respects: we consider the simultaneous substitution of multiple locations on both processes and local views. If σ is a substitution, $\sigma\{\eta_1 \mapsto \eta_2\}$ denotes the substitution that maps η_1 to η_2 and behaves like σ elsewhere.

Definition A.2 (tpl-substitution). We say that a location substitution σ is a *type preserving location substitution* (tpl-substitution) from a local view Γ to the local view Δ if

$$\forall \eta.\Gamma(\eta) \preceq \Delta(\eta\sigma).$$

Lemma A.3 (Location substitution). If we have
— σ is a tpl-substitution from Γ to Δ, and
— $\Sigma \vdash_c P : \Gamma$,
then $\Sigma \vdash_c P\sigma : \Delta$.

Proof. The proof is by structural induction on processes. It is easy to prove most cases by applying the induction hypothesis on the premises of each rule. For the rules (Recv) and (Send), observe that if $\Gamma(\eta) = \tilde{\varphi}$, then $\Delta(\eta\sigma) = \tilde{\varphi}$.

For rule (Amb), note that location substitution does not go recursively inside the ambient, but stops there instead. Therefore, the substitution does not change the original process. Additionally, we get $\Gamma'(\uparrow c) \preceq \Gamma(\downarrow c') \preceq \Delta(\downarrow c'\sigma) = \Delta(\downarrow c')$ (since, by definition, location substitution never affects port names), so the ambient can also be typed with Δ.

The rule (CapC) applies for inC and outC prefixes. Both cases are analogous, so we will just show the case inC. Therefore, we have the process $\mathsf{inC}(v\colon\tilde{\varphi})\alpha.P$ where (CapC)

$$\text{(CapC)}\quad \frac{\Sigma \vdash_c P : \Gamma, \tilde{\varphi}^{\uparrow v} \qquad \Sigma \vdash \alpha : \mathsf{amb}}{\Sigma \vdash_c \mathsf{inC}(v\colon\tilde{\varphi})\alpha.P : \Gamma}$$

Let $\uparrow u$ be a fresh location. So, by construction, the substitution $\sigma\{\uparrow u/\uparrow v\}$ is a tpl-substitution from $\Gamma, \tilde{\varphi}^{\uparrow v}$ to $\Delta, \tilde{\varphi}^{\uparrow u}$. Now, by the induction hypothesis, we get

$$\Sigma \vdash_c P\sigma\{\uparrow u/\uparrow v\} : \Delta, \tilde{\varphi}^{\uparrow u}$$

and hence

$$\Sigma \vdash_c \mathsf{inC}(u\colon\tilde{\varphi})\alpha.(P\sigma\{\uparrow u/\uparrow v\}) : \Delta$$

is derivable. Note that the equality

$$\mathsf{inC}(u\colon\tilde{\varphi})\alpha.(P\sigma\{\uparrow u/\uparrow v\}) = (\mathsf{inC}(v\colon\tilde{\varphi})\alpha.P)\sigma$$

holds (due to α-equivalence), so the induction step follows. □

Lemma A.4 (Local view subsumption). If $\Sigma \vdash_c P : \Gamma$ and $\Gamma \subset \Delta$, then $\Sigma \vdash_c P : \Delta$.

Lemma A.5 (Weakening). If $\Sigma \vdash_c P : \Gamma$ and $\Sigma \subset \Pi$, then $\Pi \vdash_c P : \Gamma$.

Definition A.6 (Variable substitution). We define the simultaneous substitution of name and capability variables with messages by structural induction:

— For names

$$\alpha\{\tilde{x} := \tilde{M}\} = \begin{cases} M_i & \text{if } \alpha = x_i \\ \alpha & \text{otherwise.} \end{cases}$$

— For capabilities

$$C\{\tilde{x} := \tilde{M}\} = \begin{cases} \mathsf{in}\ (\alpha\{\tilde{x} := \tilde{M}\}) & \text{if } C = \mathsf{in}\ \alpha \\ \mathsf{out}\ (\alpha\{\tilde{x} := \tilde{M}\}) & \text{if } C = \mathsf{out}\ \alpha \\ (D\{\tilde{x} := \tilde{M}\}).(E\{\tilde{x} := \tilde{M}\}) & \text{if } C = D.E \\ M_i & \text{if } C = x_i \\ C & \text{otherwise.} \end{cases}$$

— For messages

$$M\{\tilde{x} := \tilde{M}\} = \begin{cases} \alpha\{\tilde{x} := \tilde{M}\} & \text{if } M = \alpha \\ C\{\tilde{x} := \tilde{M}\} & \text{if } M = C. \end{cases}$$

— For processes

$$P\{\tilde{x} := \tilde{M}\} = \begin{cases} \mathbf{0} & \text{if } P = \mathbf{0} \\ P_1\{\tilde{x} := \tilde{M}\} \mid P_2\{\tilde{x} := \tilde{M}\} & \text{if } P = P_1 \mid P_2 \\ (\boldsymbol{\nu} n)P_1\{\tilde{x} := \tilde{M}\} & \text{if } P = (\boldsymbol{\nu} n)P_1 \\ & \text{and } n \notin \text{fn}(\tilde{M}) \\ !(P_1\{\tilde{x} := \tilde{M}\}) & \text{if } P = !P_1 \\ (\tilde{y} : \tilde{\varphi})^\eta.(P_1\{\tilde{x} := \tilde{M}\}) & \text{if } P = (\tilde{y} : \tilde{\varphi})^\eta.P_1 \\ & \text{and } (\text{fv}(\tilde{M}) \cup \tilde{x}) \cap \tilde{y} = \varnothing \\ \langle \dots, M_i\{\tilde{x} := \tilde{M}\}, \dots\rangle^\eta.(P_1\{\tilde{x} := \tilde{M}\}) & \text{if } P = \langle \dots, M_i, \dots\rangle^\eta.P_1 \\ (C\{\tilde{x} := \tilde{M}\}).(P_1\{\tilde{x} := \tilde{M}\}) & \text{if } P = C.P_1 \\ \mathsf{in/outC}(v{:}\,\tilde{\varphi})(\alpha\{\tilde{x} := \tilde{M}\}).P_1\{\tilde{x} := \tilde{M}\} & \text{if } P = \mathsf{in/outC}(v{:}\,\tilde{\varphi})\alpha.P_1 \\ \overline{\mathsf{in/outC}}(v : \tilde{\varphi}).P_1\{\tilde{x} := \tilde{M}\} & \text{if } P = \overline{\mathsf{in/outC}}(v : \tilde{\varphi}).P_1 \\ \alpha\{\tilde{x} := \tilde{M}\}[\,\Gamma \parallel c \parallel P_1\{\tilde{x} := \tilde{M}\}\,] & \text{if } P = \alpha[\,\Gamma \parallel c \parallel P_1\,]. \end{cases}$$

Lemma A.7. Let $\Sigma = \Sigma', x_1 : \varphi_1, \dots, x_k : \varphi_k$ and $\Sigma' \vdash M_i : \varphi_i$ for $i \in [1 \dots k]$.

1 If $\Sigma \vdash M' : \varphi'$, then $\Sigma' \vdash M'\{\tilde{x} := \tilde{M}\} : \varphi'$.

2 if $\Sigma \vdash_c P : \Gamma$, then $\Sigma' \vdash_c P\{\tilde{x} := \tilde{M}\} : \Gamma$.

Proof. The proofs of both parts are by induction on the height of the derivation. We will just consider two interesting cases:

(Amb)

$$\frac{\Sigma \vdash_{c'} P : \Gamma' \quad \Sigma \vdash \alpha : \mathsf{amb} \quad \Gamma'(\uparrow c) \preceq \Gamma(\downarrow c') \quad \Gamma' \text{ is closed}}{\Sigma \vdash_c \alpha[\,\Gamma' \parallel c' \parallel P\,] : \Gamma} \text{ (Amb)}$$

By the induction hypothesis, $\Sigma' \vdash_{c'} P\{\tilde{x} := \tilde{M}\} : \Gamma'$ and $\Sigma' \vdash \alpha\{\tilde{x} := \tilde{M}\} : \mathsf{amb}$ hold. Then, by (Amb) and the definition of substitution, we derive $\Sigma' \vdash_c \alpha[\,\Gamma' \parallel c' \parallel P\,]\{\tilde{x} := \tilde{M}\} : \Gamma$, and the induction step follows.

(Recv)

$$\frac{\Sigma, y_1 : \varphi_1, \dots, y_k : \varphi_k \vdash_c P : \Gamma \quad \Gamma(\eta) = (\varphi_1, \dots, \varphi_k)}{\Sigma \vdash_c (y_1 : \varphi_1, \dots, y_k : \varphi_k)^\eta.P : \Gamma} \text{ (Recv)}$$

By α-renaming, we can assume $(\text{fv}(\tilde{M}) \cup \tilde{x}) \cap \tilde{y} = \varnothing$. By the induction hypothesis, we derive

$$\Sigma', y_1 : \varphi_1, \dots, y_k : \varphi_k \vdash_c P\{\tilde{x} := \tilde{M}\} : \Gamma,$$

and hence by rule (Recv),

$$\Sigma' \vdash_c (y_1 : \varphi_1, \dots, y_k : \varphi_k)^\eta.(P\{\tilde{x} := \tilde{M}\}) : \Gamma.$$

Finally, by the definition of substitution,

$$\Sigma' \vdash_c ((y_1 : \varphi_1, \dots, y_k : \varphi_k)^\eta.P)\{\tilde{x} := \tilde{M}\} : \Gamma,$$

and the induction step follows. □

We can now prove subject reduction.

Theorem 2.1 (Subject Reduction). If $\Sigma \vdash_c P : \Gamma$ and $P \longrightarrow Q$, then $\Sigma \vdash_c Q : \Gamma$.

Proof. The proof is by induction on the definition of $\longrightarrow$. The most interesting cases are the base cases, as it is easy to prove the induction cases by applying Lemma A.1 and the induction hypothesis.

Communication reductions are proved using Lemma A.7. We consider two paradigmatic cases of movements.

(Red-Enter) We have

$$P \longrightarrow Q$$

with

$$\begin{aligned} P &= n[\Gamma_n \parallel c_n \parallel \mathsf{in}\ m.P_1 \mid P_2] \mid m[\Gamma_m \parallel c_m \parallel \overline{\mathsf{in}}.Q_1 \mid Q_2], \\ Q &= m[\Gamma_m \parallel c_m \parallel n[\Gamma_n \parallel c_n \parallel P_1 \mid P_2] \mid Q_1 \mid Q_2] \end{aligned}$$

where $\Gamma_n(\uparrow c_m) \preceq \Gamma_m(\downarrow c_n)$.

For this reduction rule, the derivation of $\Sigma \vdash_c P : \Gamma$ ends in

$$\dfrac{\Sigma \vdash_c n[\Gamma_n \parallel c_n \parallel \mathsf{in}\ m.P_1 \mid P_2] : \Gamma \quad \Sigma \vdash_c m[\Gamma_m \parallel c_m \parallel \overline{\mathsf{in}}.Q_1 \mid Q_2] : \Gamma}{\Sigma \vdash_c n[\Gamma_n \parallel c_n \parallel \mathsf{in}\ m.P_1 \mid P_2] \ \mid\ m[\Gamma_m \parallel c_m \parallel \overline{\mathsf{in}}.Q_1 \mid Q_2] : \Gamma}\ (\textsc{Comp})$$

The derivations above end in

$$\dfrac{\dfrac{\dfrac{\Sigma \vdash_{c_n} P_1 : \Gamma_n}{\Sigma \vdash_{c_n} \mathsf{in}\ m.P_1 : \Gamma_n}\ (\textsc{Cap}) \quad \Sigma \vdash_{c_n} P_2 : \Gamma_n}{\Sigma \vdash_{c_n} \mathsf{in}\ m.P_1 \mid P_2 : \Gamma_n}\ (\textsc{Comp}) \quad \Sigma \vdash n : \mathsf{amb} \quad \Gamma_n(\uparrow c) \preceq \Gamma(\downarrow c_n) \quad \Gamma_n \text{ is closed}}{\Sigma \vdash_c n[\Gamma_n \parallel c_n \parallel \mathsf{in}\ m.P_1 \mid P_2] : \Gamma}\ (\textsc{Amb})$$

and

$$\dfrac{\dfrac{\dfrac{\Sigma \vdash_{c_m} Q_1 : \Gamma_m}{\Sigma \vdash_{c_m} \overline{\mathsf{in}}.Q_1 : \Gamma_m}\ (\textsc{coCapC}) \quad \Sigma \vdash_{c_m} Q_2 : \Gamma_m}{\Sigma \vdash_{c_m} \overline{\mathsf{in}}.Q_1 \mid Q_2 : \Gamma_m}\ (\textsc{Comp}) \quad \Sigma \vdash m : \mathsf{amb} \quad \Gamma_m(\uparrow c) \preceq \Gamma(\downarrow c_m) \quad \Gamma_m \text{ is closed}}{\Sigma \vdash_c m[\Gamma_m \parallel c_m \parallel \overline{\mathsf{in}}.Q_1 \mid Q_2] : \Gamma}\ (\textsc{Amb})$$

respectively.

From the previous derivations and the hypothesis we obtain

$$\dfrac{\dfrac{\Sigma \vdash_{c_n} P_1 : \Gamma_n \quad \Sigma \vdash_{c_n} P_2 : \Gamma_n}{\Sigma \vdash_{c_n} P_1 \mid P_2 : \Gamma_n}\ (\textsc{Comp}) \quad \Sigma \vdash n : \mathsf{amb} \quad \Gamma_n(\uparrow c_m) \preceq \Gamma_m(\downarrow c_n) \quad \Gamma_n \text{ is closed}}{\Sigma \vdash_{c_m} n[\Gamma_n \parallel c_n \parallel P_1 \mid P_2] : \Gamma_m}\ (\textsc{Amb})$$

and

$$\frac{\Sigma \vdash_{c_m} Q_1 : \Gamma_m \quad \Sigma \vdash_{c_m} Q_2 : \Gamma_m}{\Sigma \vdash_{c_m} Q_1 \mid Q_2 : \Gamma_m} \text{ (Comp)}$$

Finally, composing these two derivations using (Comp) we get

$$\cfrac{\cfrac{\begin{array}{c}\Sigma \vdash_{c_m} n[\![\Gamma_n \parallel c_n \parallel P_1 \mid P_2]\!] : \Gamma_m \\ \Sigma \vdash_{c_m} Q_1 \mid Q_2 : \Gamma_m\end{array}}{\Sigma \vdash_{c_m} n[\![\Gamma_n \parallel c_n \parallel P_1 \mid P_2]\!] \mid Q_1 \mid Q_2 : \Gamma_m} \text{ (Comp)} \\ \Sigma \vdash m : \mathsf{amb} \qquad \Gamma_m(\uparrow c) \preceq \Gamma(\downarrow c_m) \qquad \Gamma_m \text{ is closed}}{m[\![\Gamma_m \parallel c_m \parallel n[\![\Gamma_n \parallel c_n \parallel P_1 \mid P_2]\!] \mid Q_1 \mid Q_2]\!]} \text{ (Amb)}$$

and the induction step follows.

(Red-EnterC) We have

$$P \longrightarrow Q$$

where

$$\begin{aligned} P &= n[\![\Gamma_n \parallel c_n \parallel \mathsf{inC}(v{:}\tilde{\varphi})m.P_1 \mid P_2]\!] \mid m[\![\Gamma_m \parallel c_m \parallel \overline{\mathsf{inC}}(v' : \tilde{\varphi}).Q_1 \mid Q_2]\!], \\ Q &= m[\![\Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n} \parallel c_m \parallel n[\![\Gamma_n \oplus \tilde{\varphi}^{\uparrow c_m} \parallel c_n \parallel P_1\{\uparrow c_m/\uparrow v\} \mid P_2]\!] \mid Q_1\{\downarrow c_n/\downarrow v'\} \mid Q_2]\!] \end{aligned}$$

where $\Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n}$ and $\Gamma_n \oplus \tilde{\varphi}^{\uparrow c_m}$ are defined.

For this reduction rule, the derivation of $\Sigma \vdash_c P : \Gamma$ ends in

$$\frac{\begin{array}{c}\Sigma \vdash_c n[\![\Gamma_n \parallel c_n \parallel \mathsf{inC}(v{:}\tilde{\varphi})m.P_1 \mid P_2]\!] : \Gamma \\ \Sigma \vdash_c m[\![\Gamma_m \parallel c_m \parallel \overline{\mathsf{inC}}(v' : \tilde{\varphi}).Q_1 \mid Q_2]\!] : \Gamma\end{array}}{\Sigma \vdash_c n[\![\Gamma_n \parallel c_n \parallel \mathsf{inC}(v{:}\tilde{\varphi})m.P_1 \mid P_2]\!] \;\mid\; m[\![\Gamma_m \parallel c_m \parallel \overline{\mathsf{inC}}(v' : \tilde{\varphi}).Q_1 \mid Q_2]\!] : \Gamma} \text{ (Comp)}$$

The derivations above end in

$$\cfrac{\cfrac{\cfrac{\Sigma \vdash_{c_n} P_1 : \Gamma_n, \tilde{\varphi}^{\uparrow v}}{\Sigma \vdash_{c_n} \mathsf{inC}(v{:}\tilde{\varphi})m.P_1 : \Gamma_n} \text{ (CapC)} \quad \Sigma \vdash_{c_n} P_2 : \Gamma_n}{\Sigma \vdash_{c_n} \mathsf{inC}(v{:}\tilde{\varphi})m.P_1 \mid P_2 : \Gamma_n} \text{ (Comp)} \\ \Sigma \vdash n : \mathsf{amb} \quad \Gamma_n(\uparrow c) \preceq \Gamma(\downarrow c_n) \quad \Gamma_n \text{ is closed}}{\Sigma \vdash_c n[\![\Gamma_n \parallel c_n \parallel \mathsf{inC}(v{:}\tilde{\varphi})m.P_1 \mid P_2]\!] : \Gamma} \text{ (Amb)}$$

and

$$\cfrac{\cfrac{\cfrac{\Sigma \vdash_{c_m} Q_1 : \Gamma_m, \tilde{\varphi}^{\downarrow v'}}{\Sigma \vdash_{c_m} \overline{\mathsf{inC}}(v' : \tilde{\varphi}).Q_1 : \Gamma_m} \text{ (coCapC)} \quad \Sigma \vdash_{c_m} Q_2 : \Gamma_m}{\Sigma \vdash_{c_m} \overline{\mathsf{inC}}(v' : \tilde{\varphi}).Q_1 \mid Q_2 : \Gamma_m} \text{ (Comp)} \\ \Sigma \vdash m : \mathsf{amb} \quad \Gamma_m(\uparrow c) \preceq \Gamma(\downarrow c_m) \quad \Gamma_m \text{ is closed}}{\Sigma \vdash_c m[\![\Gamma_m \parallel c_m \parallel \overline{\mathsf{inC}}(v' : \tilde{\varphi}).Q_1 \mid Q_2]\!] : \Gamma} \text{ (Amb)}$$

respectively.

Note that $\Gamma'_m = \Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n}$ and $\Gamma'_n = \Gamma_n \oplus \tilde{\varphi}^{\uparrow c_m}$ are defined by hypothesis, and the substitutions $\{\downarrow c_n/\downarrow v'\}$ and $\{\uparrow c_m/\uparrow v\}$ are tpl-substitutions from $\Gamma_m, \tilde{\varphi}^{\downarrow v'}$ to Γ'_m and from $\Gamma_n, \tilde{\varphi}^{\downarrow v}$ to Γ'_n, respectively.

Applying Lemma A.3 to $\Sigma \vdash_{c_n} P_1 : \Gamma_n, \tilde{\varphi}^{\uparrow v}$ and $\Sigma \vdash_{c_m} Q_1 : \Gamma_m, \tilde{\varphi}^{\downarrow v'}$, respectively, and Lemma A.4 to $\Sigma \vdash_{c_n} P_2 : \Gamma_n$ and $\Sigma \vdash_{c_m} Q_2 : \Gamma_m$, respectively, we can derive

$$\dfrac{\dfrac{\Sigma \vdash_{c_n} P_1\{\uparrow c_m/\uparrow v\} : \Gamma_n \oplus \tilde{\varphi}^{\uparrow c_m} \qquad \Sigma \vdash_{c_n} P_2 : \Gamma_n \oplus \tilde{\varphi}^{\uparrow c_m}}{\Sigma \vdash_{c_n} P_1\{\uparrow c_m/\uparrow v\} \mid P_2 : \Gamma_n \oplus \tilde{\varphi}^{\uparrow c_m}}\ (\textsc{Comp}) \quad \Sigma \vdash n : \mathsf{amb} \quad \Gamma_n(\uparrow c_m) \oplus \tilde{\varphi}^{\uparrow c_m} \preceq \Gamma_m(\downarrow c_n) \oplus \tilde{\varphi}^{\downarrow c_n} \quad \Gamma_n \oplus \tilde{\varphi}^{\uparrow c_m} \text{ is closed}}{\Sigma \vdash_{c_m} n[\Gamma_n \oplus \tilde{\varphi}^{\uparrow c_m} \parallel c_n \parallel P_1\{\uparrow c_m/\uparrow v\} \mid P_2] : \Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n}}\ (\textsc{Amb})$$

and

$$\dfrac{\Sigma \vdash_{c_m} Q_1\{\downarrow c_n/\downarrow v'\} : \Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n} \quad \Sigma \vdash_{c_m} Q_2 : \Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n}}{\Sigma \vdash_{c_m} Q_1\{\downarrow c_n/\downarrow v'\} \mid Q_2 : \Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n}}\ (\textsc{Comp})$$

Finally, composing these two derivations using (Comp), we get

$$\dfrac{\dfrac{\begin{array}{c}\Sigma \vdash_{c_m} n[\Gamma_n \oplus \tilde{\varphi}^{\uparrow c_m} \parallel c_n \parallel P_1\{\uparrow c_m/\uparrow v\} \mid P_2] : \Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n} \\ \Sigma \vdash_{c_m} Q_1\{\downarrow c_n/\downarrow v'\} \mid Q_2 : \Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n}\end{array}}{\Sigma \vdash_{c_m} n[\Gamma_n \oplus \tilde{\varphi}^{\uparrow c_m} \parallel c_n \parallel P_1\{\uparrow c_m/\uparrow v\} \mid P_2] \mid Q_1\{\downarrow c_n/\downarrow v'\} \mid Q_2 : \Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n}}\ (\textsc{Comp}) \quad \Sigma \vdash m : \mathsf{amb} \quad \Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n}(\uparrow c) = \Gamma_m(\uparrow c) \preceq \Gamma(\downarrow c_m) \quad \Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n} \text{ is closed}}{m[\Gamma_m \oplus \tilde{\varphi}^{\downarrow c_n} \parallel c_m \parallel n[\Gamma_n \oplus \tilde{\varphi}^{\uparrow c_m} \parallel c_n \parallel P_1\{\uparrow c_m/\uparrow v\} \mid P_2] \mid Q_1\{\downarrow c_n/\downarrow v'\} \mid Q_2]}\ (\textsc{Amb})$$

□

Appendix B. Independence from barbs (Theorem 4.4)

Lemma B.1. If P is a well-formed process in the environment Σ, then for all c there exists Γ such that $\Sigma \vdash_c P : \Gamma$.

Proof. We can build Γ by structural induction on P as the result of the map $\mathscr{LV}(\Sigma, P, c)$ defined by:

$$\begin{array}{rcl}
\mathscr{LV}(\Sigma, \mathbf{0}, c) & = & \varnothing \\
\mathscr{LV}(\Sigma, P_1 \mid P_2, c) & = & \mathscr{LV}(\Sigma, P_1, c) \cup \mathscr{LV}(\Sigma, P_2, c) \\
\mathscr{LV}(\Sigma, (\boldsymbol{\nu} n)P', c) & = & \mathscr{LV}(\Sigma, P', c) \\
\mathscr{LV}(\Sigma, !P', c) & = & \mathscr{LV}(\Sigma, P', c) \\
\mathscr{LV}(\Sigma, \pi.P', c) & = & \begin{cases}
\mathscr{LV}(\Sigma, P', c) \cup \{\tilde{\varphi}^{\eta}\} & \text{if either } \pi = (\tilde{x} : \tilde{\varphi})^{\eta} \\
 & \text{or } \pi = \langle \tilde{M} \rangle^{\eta} \text{ and } \Sigma \vdash \tilde{M} : \tilde{\varphi} \\
\mathscr{LV}(\Sigma, P', c) - \{\tilde{\varphi}^{\uparrow v}\} & \text{if either } \pi = \mathsf{inC}(v{:}\,\tilde{\varphi})\alpha \\
 & \text{or } \pi = \mathsf{outC}(v{:}\,\tilde{\varphi})\alpha \\
\mathscr{LV}(\Sigma, P', c) - \{\tilde{\varphi}^{\downarrow v}\} & \text{if either } \pi = \overline{\mathsf{inC}}(v : \tilde{\varphi}) \\
 & \text{or } \pi = \overline{\mathsf{outC}}(v : \tilde{\varphi}) \\
\mathscr{LV}(\Sigma, P', c) & \text{otherwise}
\end{cases}
\end{array}$$

$$\mathscr{LV}(\Sigma, \alpha[\![\,\Gamma_\alpha \parallel c_\alpha \parallel P'\,]\!], c) \;=\; \begin{cases} \{\tilde{\varphi}^{\downarrow c_\alpha}\} & \text{if } \Gamma_\alpha(\uparrow c) = \tilde{\varphi} \\ \varnothing & \text{otherwise.} \end{cases}$$

Note that, given Σ and $\tilde{M}$, there is a unique $\tilde{\varphi}$ such that $\Sigma \vdash \tilde{M} : \tilde{\varphi}$.
It is easy to verify that $\Sigma \vdash_c P : \mathscr{LV}(\Sigma, P, c)$, and that $\mathscr{LV}(\Sigma, P, c) \subseteq \Gamma'$ for all Γ' such that $\Sigma \vdash_c P : \Gamma'$. □

Theorem 4.4 (Independence from barbs). $\cong_i \,=\, \cong_j$ for all $i, j \in [1 \dots 4]$.

Proof. We need to show that all barbs imply each other. This can be accomplished, as usual, by exhibiting a corresponding typed context.

Suppose we have to prove that $\cong_i \subseteq \cong_j$. We need to prove that if $P \cong_i Q$, then P and Q expose the same barbs of the form j.

Following the standard procedure, we prove that for each pair $i \neq j \in [1 \dots 4]$ we can build a context $C_{i,j}[\cdot]$ such that, if $C_{i,j}[P]$ is well formed, then $C_{i,j}[P] \Downarrow^i_{\Xi}$ if and only if $P \Downarrow^j_{\Xi'}$. The name exposed in the i barb must be a fresh name introduced by the context $C_{i,j}[\cdot]$, so it cannot be exposed by the process P.

Next, we provide the details for each case:

— $P \cong_1 Q \implies P \cong_2 Q$
Assuming that P exposes the barb $\downarrow^2_{(n)}$, we need to show that Q exposes the barb $\Downarrow^2_{(n)}$. To this end, we build the context

$$C_{2,1}[\cdot] \triangleq l[\![\, \varnothing \parallel c_l \parallel \mathsf{inC}(v\!:\tilde{\varphi})n.\mathsf{out}\ n.\overline{\mathsf{in}} \,]\!] \mid [\cdot] \mid \overline{\mathsf{out}}$$

where l is a fresh name and c_l is arbitrary.
Note that for an arbitrary processes R, we have that $C_{2,1}[R]$ exposes the barb $\Downarrow^1_{(l)}$ if and only if the process R allows ambient l to enter n with a $\mathsf{inC}(v\!:\tilde{\varphi})n$ prefix. This means that R exposes the barb $\Downarrow^2_{(n)}$. Therefore, if $C_{2,1}[P] \cong_1 C_{2,1}[Q]$, then either they both expose the barb $\Downarrow^1_{(l)}$ or neither does, and hence, if P exposes the above barb, then Q must also expose the same barb.

— $P \cong_2 Q \implies P \cong_3 Q$
Assuming that P exposes the barb $\downarrow^3_{(c,c_n)}$, that $\Gamma = \mathscr{LV}(\varnothing, P, c)$ (the definition of $\mathscr{LV}(\varnothing, P, c)$ is given in the proof of Lemma B.1) and that $\Gamma(\downarrow c) = \tilde{\varphi}$, we build the context

$$C_{3,2}[\cdot] \triangleq l[\![\, \Gamma \parallel c \parallel \langle \tilde{M} \rangle^{\downarrow c_n}.\overline{\mathsf{inC}}(v : \mathsf{amb}) \mid [\cdot] \,]\!]$$

where l is a fresh name, and $\vdash \tilde{M} : \tilde{\varphi}$. Note that each type in $\tilde{\varphi}$ must be either amb or cap, so such an $\tilde{M}$ always exists.

— $P \cong_3 Q \implies P \cong_4 Q$
Assuming that P exposes the barb $\downarrow^4_{(c,c_n)}$, that $\Gamma = \mathscr{LV}(\varnothing, P, c)$ and that $\Gamma(\downarrow c) = \tilde{\varphi}$, we build the context

$$C_{4,3}[\cdot] \triangleq l[\![\, \Gamma' \parallel c \parallel (\tilde{x} : \tilde{\varphi})^{\downarrow c_n}.(y : \mathsf{amb})^{\uparrow c'} \mid [\cdot] \,]\!]$$

where $\Gamma' = \{\Gamma, \mathsf{amb}^{\uparrow c'}\}$ and l, c' are fresh.

— $P \cong_4 Q \implies P \cong_1 Q$
Assuming that P exposes the barb $\downarrow^1_{(n)}$, we build the context

$$C_{1,4}[\cdot] \triangleq l[\![\{\mathsf{amb}^{\uparrow c}\} \parallel c_l \parallel \mathsf{in}\ n.\mathsf{out}\ n.\langle l\rangle^{\uparrow c}]\!] \mid [\cdot]$$

where l, c and c_l are fresh. □

Appendix C. Observability and the labelled transition system (Lemma 4.5)

We need to relate the structure of the processes and the outcomes in the LTS rules to the labels of the visible transitions (Lemmas C.1–C.3). To this end, we extend the structural congruence for processes to concretions by adding the following axioms and rules:

(Struct Concr P)	$P \equiv P'$ and $Q \equiv Q'$	$\implies$	$(\boldsymbol{\nu}\tilde{p})\langle\!\langle P\rangle\!\rangle Q \equiv (\boldsymbol{\nu}\tilde{p})\langle\!\langle P'\rangle\!\rangle Q'$	
(Struct Concr M)	$P \equiv P'$	$\implies$	$(\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M}\rangle\!\rangle P \equiv (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M}\rangle\!\rangle P'$	
(Struct ConcrNCommP)	$(\boldsymbol{\nu}\tilde{r},\tilde{p})\langle\!\langle P\rangle\!\rangle Q$	$\equiv$	$(\boldsymbol{\nu}\tilde{p},\tilde{r})\langle\!\langle P\rangle\!\rangle Q$	
(Struct ConcrNCommM)	$(\boldsymbol{\nu}\tilde{r},\tilde{p})\langle\!\langle \tilde{M}\rangle\!\rangle P$	$\equiv$	$(\boldsymbol{\nu}\tilde{p},\tilde{r})\langle\!\langle \tilde{M}\rangle\!\rangle P$	
(Struct ConcrNLeft)	$(\boldsymbol{\nu}\tilde{p})\langle\!\langle P\rangle\!\rangle Q$	$\equiv$	$\langle\!\langle(\boldsymbol{\nu}\tilde{p})P\rangle\!\rangle\ Q$	if $\tilde{p} \cap \mathrm{fn}(Q) = \varnothing$
(Struct ConcrNRightP)	$(\boldsymbol{\nu}\tilde{p})\langle\!\langle P\rangle\!\rangle Q$	$\equiv$	$\langle\!\langle P\rangle\!\rangle\ (\boldsymbol{\nu}\tilde{p})Q$	if $\tilde{p} \cap \mathrm{fn}(P) = \varnothing$
(Struct ConcrNRightM)	$(\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M}\rangle\!\rangle Q$	$\equiv$	$\langle\!\langle \tilde{M}\rangle\!\rangle\ (\boldsymbol{\nu}\tilde{p})Q$	if $\tilde{p} \cap \mathrm{fn}(\tilde{M}) = \varnothing$

We also use the following notational conventions:

— $(\boldsymbol{\nu}\tilde{q})((\boldsymbol{\nu}\tilde{p})\langle\!\langle P\rangle\!\rangle Q)$ is defined to be $(\boldsymbol{\nu}\tilde{q},\tilde{p})\langle\!\langle P\rangle\!\rangle Q$,
— $(\boldsymbol{\nu}\tilde{q})((\boldsymbol{\nu}\tilde{p})\langle\!\langle M\rangle\!\rangle P)$ is defined to be $(\boldsymbol{\nu}\tilde{q},\tilde{p})\langle\!\langle M\rangle\!\rangle P$,
— $((\boldsymbol{\nu}\tilde{p})\langle\!\langle P\rangle\!\rangle Q) \mid R$ is defined to be $(\boldsymbol{\nu}\tilde{p})\langle\!\langle P\rangle\!\rangle (Q \mid R)$,
— $((\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M}\rangle\!\rangle P) \mid R$ is defined to be $(\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M}\rangle\!\rangle (P \mid R)$,

where $\mathrm{fn}(R) \cap \tilde{p} = \varnothing$ in the last two cases.

The proofs of the following three lemmas by induction on the LTS rules are standard.

Lemma C.1 (Structure of the processes on visible transitions – communication).

1 If $P \xrightarrow{(\tilde{M})^\eta} P'$, then for some $\tilde{\varphi}, \tilde{p}, \tilde{x}, P_1$ and Q:
$\vdash \tilde{M} : \tilde{\varphi}$,
$P \equiv (\boldsymbol{\nu}\tilde{p})((\tilde{x} : \tilde{\varphi})^\eta.P_1 \mid Q)$
and
$P' \equiv (\boldsymbol{\nu}\tilde{p})(P_1\{\tilde{x} := \tilde{M}\} \mid Q)$.

2 If $P \xrightarrow{\langle - \rangle^\eta} O$, then for some $\tilde{M}, \tilde{\varphi}, \tilde{p}, P_1$ and Q:
$\vdash \tilde{M} : \tilde{\varphi}$,
$P \equiv (\boldsymbol{\nu}\tilde{p})(\langle \tilde{M}\rangle^\eta.P_1 \mid Q)$
and
$O \equiv (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M}\rangle\!\rangle P_1 \mid Q$.

3 If $P \xrightarrow{\mathsf{get}(\tilde{M},c_n,c_m)} P'$, then for some $\tilde{\varphi}, \tilde{p}, m, \Gamma_m, \tilde{x}, P_1, P_2$ and Q:
$\vdash \tilde{M} : \tilde{\varphi}$,
$P \equiv (\boldsymbol{\nu}\tilde{p})(m[\![\Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi})^{\uparrow c_n}.P_1 \mid P_2]\!] \mid Q)$

and
$P' \equiv (\boldsymbol{\nu}\tilde{p})(m[\![\,\Gamma_m \parallel c_m \parallel P_1\{\tilde{x} := \tilde{M}\} \mid P_2\,]\!] \mid Q)$.

4 If $P \xrightarrow{\mathsf{put}(c_n,c_m)} O$, then for some $\tilde{M}, \tilde{\varphi}, \tilde{p}, m, \Gamma_m, P_1, P_2$ and Q:
$\vdash \tilde{M} : \tilde{\varphi}$,
$P \equiv (\boldsymbol{\nu}\tilde{p})(m[\![\,\Gamma_m \parallel c_m \parallel \langle \tilde{M} \rangle^{\uparrow c_n}.P_1 \mid P_2\,]\!] \mid Q)$
and
$O \equiv (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle m[\![\,\Gamma_m \parallel c_m \parallel P_1 \mid P_2\,]\!] \mid Q$.

5 If $P \xrightarrow{\mathsf{pre\text{-}comm}(c_n)} P'$, then for some $\tilde{M}, \tilde{\varphi}, \tilde{p}, m, \Gamma_m, c_m, \tilde{x}, P_1, P_2, Q_1$ and Q_2:
$\vdash \tilde{M} : \tilde{\varphi}$,
and either
$P \equiv (\boldsymbol{\nu}\tilde{p})(m[\![\,\Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi})^{\uparrow c_n}.P_1 \mid P_2\,]\!] \mid \langle \tilde{M} \rangle^{\downarrow c_m}.Q_1 \mid Q_2)$
and
$P' \equiv (\boldsymbol{\nu}\tilde{p})(m[\![\,\Gamma_m \parallel c_m \parallel P_1\{\tilde{x} := \tilde{M}\} \mid P_2\,]\!] \mid Q_1 \mid Q_2)$
or
$P \equiv (\boldsymbol{\nu}\tilde{p})(m[\![\,\Gamma_m \parallel c_m \parallel \langle \tilde{M} \rangle^{\uparrow c_n}.P_1 \mid P_2\,]\!] \mid (\tilde{x} : \tilde{\varphi})^{\downarrow c_m}.Q_1 \mid Q_2)$
and
$P' \equiv (\boldsymbol{\nu}\tilde{p})(m[\![\,\Gamma_m \parallel c_m \parallel P_1 \mid P_2\,]\!] \mid Q_1\{\tilde{x} := \tilde{M}\} \mid Q_2)$.

Lemma C.2 (Structure of the processes on visible transitions – mobility without port exchanges).

1 If $P \xrightarrow{\zeta} P'$ where $\zeta \in \{\mathsf{in}\ n, \mathsf{out}\ n, \overline{\mathsf{in}}, \overline{\mathsf{out}}\}$, then for some $\tilde{p}, P_1$ and Q:
$P \equiv (\boldsymbol{\nu}\tilde{p})(\zeta.P_1 \mid Q)$,
$P' \equiv (\boldsymbol{\nu}\tilde{p})(P_1 \mid Q)$
and
$n \notin \tilde{p}$.

2 If $P \xrightarrow{\zeta(c:\rho,c_m)n} O$ where $\zeta \in \{\mathsf{in}, \mathsf{out}\}$, then for some $\tilde{p}, m, \Gamma_m, P_1, P_2$ and Q:
$P \equiv (\boldsymbol{\nu}\tilde{p})(m[\![\,\Gamma_m \parallel c_m \parallel \zeta\ n.P_1 \mid P_2\,]\!] \mid Q)$,
$O \equiv (\boldsymbol{\nu}\tilde{p})\langle\!\langle m[\![\,\Gamma_m \parallel c_m \parallel P_1 \mid P_2\,]\!] \rangle\!\rangle Q$,
$\Gamma_m(\uparrow c) = \rho$
and
$n \notin \tilde{p}$.

3 If $P \xrightarrow{[\![\,\Gamma_n \parallel c_n \parallel \overline{\mathsf{in}}\,n\,]\!]} O$, then for some $\tilde{p}, P_1, P_2$ and Q:
$P \equiv (\boldsymbol{\nu}\tilde{p})(n[\![\,\Gamma_n \parallel c_n \parallel \overline{\mathsf{in}}.P_1 \mid P_2\,]\!] \mid Q)$,
$O \equiv (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \mid P_2 \rangle\!\rangle Q$
and
$n \notin \tilde{p}$.

4 If $P \xrightarrow{\mathsf{pop}(c:\rho,c_m)} P'$, then for some $\tilde{p}, n, \Gamma_n, c_n, m, \Gamma_m, P_1, P_2, Q_1$ and Q_2:
$P \equiv (\boldsymbol{\nu}\tilde{p})(n[\![\,\Gamma_n \parallel c_n \parallel m[\![\,\Gamma_m \parallel c_m \parallel \mathsf{out}\ n.P_1 \mid P_2\,]\!] \mid Q_1\,]\!] \mid Q_2)$,
$P' \equiv (\boldsymbol{\nu}\tilde{p})(n[\![\,\Gamma_n \parallel c_n \parallel Q_1\,]\!] \mid Q_2 \mid m[\![\,\Gamma_m \parallel c_m \parallel P_1 \mid P_2\,]\!])$
and
$\Gamma_m(\uparrow c) = \rho$.

5 If $P \xrightarrow{\mathsf{pre\text{-}exit}(c:\rho,c_m)} P'$, then for some $\tilde{p}, n, \Gamma_n, c_n, m, \Gamma_m, P_1, P_2, Q_1, Q_2$ and Q_3:
$P \equiv (\boldsymbol{\nu}\tilde{p})(n[\![\, \Gamma_n \parallel c_n \parallel m[\![\, \Gamma_m \parallel c_m \parallel \mathsf{out}\ n.P_1 \mid P_2 \,]\!] \mid Q_1 \,]\!] \mid \overline{\mathsf{out}}.Q_2 \mid Q_3)$,
$P' \equiv (\boldsymbol{\nu}\tilde{p})(n[\![\, \Gamma_n \parallel c_n \parallel Q_1 \,]\!] \mid m[\![\, \Gamma_m \parallel c_m \parallel P_1 \mid P_2 \,]\!] \mid Q_2 \mid Q_3)$
and
$\Gamma_m(\uparrow c) = \rho$.

Lemma C.3 (Structure of the processes on visible transitions – mobility with port exchanges).

1 If $P \xrightarrow{\zeta(c\,:\,\tilde{\varphi})n} P'$ where $\zeta \in \{\mathsf{inC}, \mathsf{outC}\}$, then for some $\tilde{p}, v, P_1$ and Q:
$P \equiv (\boldsymbol{\nu}\tilde{p})(\zeta(v\!:\tilde{\varphi})n.P_1 \mid Q)$,
$P' \equiv (\boldsymbol{\nu}\tilde{p})(P_1\{\uparrow c/\uparrow v\} \mid Q)$
and
$n \notin \tilde{p}$.

2 If $P \xrightarrow{\overline{\zeta}(c_m\,:\,\tilde{\varphi})} P'$ where $\overline{\zeta} \in \{\overline{\mathsf{inC}}, \overline{\mathsf{outC}}\}$, then for some $\tilde{p}, v, P_1$ and Q:
$P \equiv (\boldsymbol{\nu}\tilde{p})(\overline{\zeta}(v : \tilde{\varphi}).P_1 \mid Q)$
and
$P' \equiv (\boldsymbol{\nu}\tilde{p})(P_1\{\downarrow c_m/\downarrow v\} \mid Q)$.

3 If $P \xrightarrow{\zeta(c:\tilde{\varphi},c_m)n} O$ where $\zeta \in \{\mathsf{inC}, \mathsf{outC}\}$, then for some $\tilde{p}, m, \Gamma_m, v, P_1, P_2$ and Q:
$P \equiv (\boldsymbol{\nu}\tilde{p})(m[\![\, \Gamma_m \parallel c_m \parallel \zeta(v : \tilde{\varphi})n.P_1 \mid P_2 \,]\!] \mid Q)$,
$O \equiv (\boldsymbol{\nu}\tilde{p})\langle\!\langle m[\![\, \Gamma_m \oplus \tilde{\varphi}^{\uparrow c} \parallel c_m \parallel P_1\{\uparrow c/\uparrow v\} \mid P_2 \,]\!]\rangle\!\rangle Q$
and
$n \notin \tilde{p}$.

4 If $P \xrightarrow{[\![\, \Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(c_m\,:\,\tilde{\varphi})n \,]\!]} O$, then for some $\tilde{p}, v, P_1, P_2$ and Q:
$P \equiv (\boldsymbol{\nu}\tilde{p})(n[\![\, \Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(v : \tilde{\varphi}).P_1 \mid P_2 \,]\!] \mid Q)$,
$O \equiv (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1\{\downarrow c_m/\downarrow v\} \mid P_2\rangle\!\rangle Q$,
$\Gamma_n \oplus \tilde{\varphi}^{\downarrow c_m}$ is defined
and
$n \notin \tilde{p}$.

5 If $P \xrightarrow{\mathsf{popC}(c\,:\,\tilde{\varphi},c_m)} P'$, then for some $\tilde{p}, n, \Gamma_n, c_n, m, \Gamma_m, v, P_1, P_2, Q_1$ and Q_2:
$P \equiv (\boldsymbol{\nu}\tilde{p})(n[\![\, \Gamma_n \parallel c_n \parallel m[\![\, \Gamma_m \parallel c_m \parallel \mathsf{outC}(v\!:\tilde{\varphi})n.P_1 \mid P_2 \,]\!] \mid Q_1 \,]\!] \mid Q_2)$,
$P' \equiv (\boldsymbol{\nu}\tilde{p})(n[\![\, \Gamma_n \parallel c_n \parallel Q_1 \,]\!] \mid m[\![\, \Gamma_m \oplus \tilde{\varphi}^{\uparrow c} \parallel c_m \parallel P_1 \mid P_2 \,]\!] \mid Q_2)$
and
$\Gamma \oplus \tilde{\varphi}^{\uparrow c}$ is defined.

6 If $P \xrightarrow{\mathsf{pre\text{-}exitC}(c\,:\,\tilde{\varphi},c_n)} P'$, then for some $\tilde{p}, n, \Gamma_n, c_n, m, \Gamma_m, v, P_1, P_2, Q_1, u, Q_2$, and Q_3:
$P \equiv (\boldsymbol{\nu}\tilde{p})(n[\![\, \Gamma_n \parallel c_n \parallel m[\![\, \Gamma_m \parallel c_m \parallel \mathsf{outC}(v\!:\tilde{\varphi})n.P_1 \mid P_2 \,]\!] \mid Q_1 \,]\!] \mid \overline{\mathsf{outC}}(u : \tilde{\varphi}).Q_2 \mid Q_3)$,
$P' \equiv (\boldsymbol{\nu}\tilde{p})(n[\![\, \Gamma_n \parallel c_n \parallel Q_1 \,]\!] \mid m[\![\, \Gamma_m \oplus \tilde{\varphi}^{\uparrow c} \parallel c_m \parallel P_1\{\uparrow c/\uparrow v\} \mid P_2 \,]\!] \mid Q_2\{\downarrow c_m/\downarrow u\} \mid Q_3)$
and
$\Gamma \oplus \tilde{\varphi}^{\downarrow c_m}$ is defined.

The following three lemmas are useful for showing that structurally congruent processes have the same labelled transitions (Lemma C.7).

Lemma C.4 (LTS message receive). If $P \xrightarrow{(\tilde{M})^\eta} P'$ and $\tilde{n} \cap \mathrm{fn}(P) = \varnothing$, then, for any fresh names $\tilde{m}$, we have $P \xrightarrow{(\tilde{M}')^\eta} P''$ where $\tilde{M}' = \tilde{M}\{\tilde{n} := \tilde{m}\}$, $P'' = P'\{\tilde{n} := \tilde{m}\}$.

Proof. We proceed by induction on the LTS rules. Most of the rules hold trivially or can be proved straightforwardly using the induction hypothesis. We only consider the proof for the rules (LTS RECV) and (LTS RES).

— (LTS RECV)
We have

$$P = (\tilde{x} : \tilde{\varphi})^\eta.P_1 \xrightarrow{(\tilde{M})^\eta} P_1\{\tilde{x} := \tilde{M}\} = P'\,.$$

So, by the same rule,

$$P \xrightarrow{(\tilde{M}')^\eta} P_1\{\tilde{x} := \tilde{M}'\} = P''$$

where $\tilde{M}' = \tilde{M}\{\tilde{n} := \tilde{m}\}$. Finally,

$$P'' = P_1\{\tilde{x} := \tilde{M}\{\tilde{n} := \tilde{m}\}\} = P'\{\tilde{n} := \tilde{m}\}$$

since $\tilde{n} \cap \mathrm{fn}(P_1) = \varnothing$ and $\tilde{m}$ are fresh.

— (LTS RES)
We have

$$P = (\boldsymbol{\nu}\tilde{p})P_1 \xrightarrow{(\tilde{M})^\eta} (\boldsymbol{\nu}\tilde{p})(P_1\{\tilde{x} := \tilde{M}\}) = P'$$

because

$$P_1 \xrightarrow{(\tilde{M})^\eta} P_1\{\tilde{x} := \tilde{M}\} = P_1'$$

and $\tilde{p} \cap \mathrm{fn}(\tilde{M}) = \varnothing$. By the induction hypothesis,

$$P_1 \xrightarrow{(\tilde{M}')^\eta} P_1'\{\tilde{n} := \tilde{m}\}$$

where $\tilde{M}' = \tilde{M}\{\tilde{n} := \tilde{m}\}$. Since $\mathrm{fn}(\tilde{M}') \subseteq \mathrm{fn}(\tilde{M}) \cup \tilde{m}$ and $\tilde{p} \cap \tilde{m} = \varnothing$, we apply the rule (LTS RES) to get

$$P \xrightarrow{(\tilde{M}')^\eta} (\boldsymbol{\nu}\tilde{p})(P_1'\{\tilde{n} := \tilde{m}\}) = P''\,.$$

Finally, note that

$$P'' = (\boldsymbol{\nu}\tilde{p})(P_1\{\tilde{x} := \tilde{M}\{\tilde{n} := \tilde{m}\}\}) = (\boldsymbol{\nu}\tilde{p})(P_1'\{\tilde{n} := \tilde{m}\})$$

because $\mathrm{fn}(P_1) \cap \tilde{n} = \varnothing$, and

$$(\boldsymbol{\nu}\tilde{p})(P_1'\{\tilde{n} := \tilde{m}\}) = P'\{\tilde{n} := \tilde{m}\}$$

because $\tilde{n}$ can only appear free in $\tilde{M}$, $\tilde{p} \cap \mathrm{fn}(\tilde{M}) = \varnothing$ and $\tilde{m}$ are fresh. □

Lemma C.5 (Location renaming and message variable substitution preserve structural equivalence). If $P \equiv Q$, then:

1 $P\{\eta'/\eta\} \equiv Q\{\eta'/\eta\}$;
2 $P\{\tilde{x} := \tilde{M}\} \equiv Q\{\tilde{x} := \tilde{M}\}$.

Proof. The proof is by induction on the structural equivalence rules. □

Lemma C.6. If $P \equiv Q$, then $\mathrm{fn}(P) = \mathrm{fn}(Q)$.

Proof. The proof is by induction on the structural equivalence rules. □

Lemma C.7. If $P \xrightarrow{\xi} O$ and $P \equiv Q$, there exists O' such that $Q \xrightarrow{\xi} O'$ and $O \equiv O'$.

Proof. As with Lemma A.1, we must show a stronger statement in order to make the proof easier. That statement is:
If
(a) $P \xrightarrow{\xi} O$ and $P \equiv Q$,
or
(b) $P \xrightarrow{\xi} O$ and $Q \equiv P$,

there exists O' such that $Q \xrightarrow{\xi} O'$.

The proof is by simultaneous induction on the derivations of $P \equiv Q$ and $Q \equiv P$. For each structural equivalence rule we need to inspect both sides of the equivalence in the consequent of the rule. For each of these subcases we have to consider different sub-subcases for each LTS rule that matches the structure of the subcases. This gives us a lot of sub-subcases, but most of them have similar proofs. For the induction cases of the structural rules, most of the proofs are tedious but simple. The most interesting cases are the base cases. Some of these use Lemmas C.4–C.6. □

Lemma 4.5 $P \downarrow^1_{(n)}$ if and only if $P \xrightarrow{[\Gamma_n \parallel c_n \parallel \overline{\mathsf{in}}\ n]} O$ for some Γ_n, c_n, and O.

Proof. By definition, in order to expose the barb n, the process P must be structurally equivalent to $(\boldsymbol{\nu}\tilde{m})(n[\Gamma_n \parallel c_n \parallel \overline{\mathsf{in}}.Q \mid R] \mid S)$. Using Lemma C.7, we can show that P moves using the transition above. The converse is easily proved by Lemma C.2(3). □

Appendix D. Coincidence between unlabelled and labelled reductions (Theorem 4.6)

Theorem 4.6. Let P be closed.

(1) If $P \xrightarrow{\tau} P'$, then $P \longrightarrow P'$.

(2) If $P \longrightarrow P'$, then $P \xrightarrow{\tau} Q$ and $Q \equiv P'$ for some Q.

Proof.

(1) We use induction on the LTS rules. For each case, we use Lemmas C.1–C.3 to determine the structure of the process P so that we can apply the reduction rules, and, hence, show that P reduces to P'. Most of the cases are similar. We illustrate, as an example, the proof for the rule (LTS τ-EnterC).

$$\text{(LTS } \tau\text{-EnterC)}\quad \frac{\begin{array}{c} P \xrightarrow{\mathsf{inC}(c_n\,:\,\tilde{\varphi},c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m\rangle\!\rangle P' \\ Q \xrightarrow{[\Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(c_m\,:\,\tilde{\varphi})n]} (\boldsymbol{\nu}\tilde{q})\langle\!\langle Q_n\rangle\!\rangle Q' \\ \mathrm{fn}(P)\cap\tilde{q} = \mathrm{fn}(Q)\cap\tilde{p} = \tilde{p}\cap\tilde{q} = \varnothing \end{array}}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p},\tilde{q})(n[\Gamma_n \oplus \tilde{\varphi}^{\downarrow c_m} \parallel c_n \parallel Q_n \mid P_m] \mid P' \mid Q')}$$

By Lemma C.3(3) and (4),
$P \equiv (\boldsymbol{\nu}\tilde{p})(m[\![\, \Gamma_m \parallel c_n \parallel \mathsf{inC}(v : \tilde{\varphi}) n.P_1 \mid P_2 \,]\!] \mid P')$,
$P_m \equiv m[\![\, \Gamma_m \oplus \tilde{\varphi}^{\uparrow c_n} \parallel c_m \parallel P_1\{\uparrow c_n / \uparrow v\} \mid P_2 \,]\!]$
and
$n \notin \tilde{p}$,
for some m, Γ_m, v, P_1 and P_2,
and
$Q \equiv (\boldsymbol{\nu}\tilde{q})(n[\![\, \Gamma'_n \parallel c_n \parallel \overline{\mathsf{inC}}(u : \tilde{\varphi}).Q_1 \mid Q_2 \,]\!] \mid Q')$,
$Q_n \equiv Q_1\{\downarrow c_m / \downarrow u\} \mid Q_2$,
$\Gamma_n \oplus \tilde{\varphi}^{\downarrow c_m}$ is defined
and
$n \notin \tilde{p}$,
for some u, Q_1 and Q_2.

Now,

$$\begin{aligned} P \mid Q \;\equiv\; & (\boldsymbol{\nu}\tilde{p})(m[\![\, \Gamma_m \parallel c_m \parallel \mathsf{inC}(v : \tilde{\varphi}) n.P_1 \mid P_2 \,]\!] \mid P') \mid \\ & (\boldsymbol{\nu}\tilde{q})(n[\![\, \Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(u : \tilde{\varphi}).Q_1 \mid Q_2 \,]\!] \mid Q') \\ \equiv\; & (\boldsymbol{\nu}\tilde{p}, \tilde{q})(m[\![\, \Gamma_m \parallel c_m \parallel \mathsf{inC}(v : \tilde{\varphi}) n.P_1 \mid P_2 \,]\!] \mid \\ & n[\![\, \Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(u : \tilde{\varphi}).Q_1 \mid Q_2 \,]\!] \mid P' \mid Q') \end{aligned}$$

because $\mathrm{fn}(Q) \cap \tilde{p} = \varnothing, \mathrm{fn}(P) \cap \tilde{q} = \varnothing$, and $\tilde{p} \cap \tilde{q} = \varnothing$.
By (Red-EnterC)

$$\begin{gathered} m[\![\, \Gamma_m \parallel c_m \parallel \mathsf{inC}(v : \tilde{\varphi}) n.P_1 \mid P_2 \,]\!] \mid n[\![\, \Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(u : \tilde{\varphi}).Q_1 \mid Q_2 \,]\!] \\ \longrightarrow \\ n[\![\, \Gamma_n \oplus \tilde{\varphi}^{\downarrow c_m} \parallel c_n \parallel Q_1\{\downarrow c_m / \downarrow u\} \mid Q_2 \mid m[\![\, \Gamma_m \oplus \tilde{\varphi}^{\uparrow c_n} \parallel c_n \parallel P_1\{\uparrow c_n / \uparrow v\} \mid P_2 \,]\!]\,]\!]. \end{gathered}$$

Finally, applying the structural reduction rules in Table 7, we conclude that

$$P \mid Q \quad \longrightarrow \quad (\boldsymbol{\nu}\tilde{p}, \tilde{q})(n[\![\, \Gamma_n \oplus \tilde{\varphi}^{\downarrow c_m} \parallel c_n \parallel Q_n \mid P_m \,]\!] \mid P' \mid Q').$$

(2) The proof of this part is routine using induction on the reduction rules. □

Appendix E. Congruence of full bisimilarity (Theorem 4.10)

In this appendix we prove the same result as we proved for first-order transitions and arbitrary outcomes in Lemma C.7, but now allowing higher-order transitions and restricting the outcomes to be processes.

Lemma E.1 ($\equiv$ is a bisimulation). If $P \xrightarrow{\lambda} P'$ and $P \equiv Q$, there exists Q' such that $Q \xrightarrow{\lambda} Q'$ and $P' \equiv Q'$.

Proof. Lemma C.7 applies directly for most of the cases. The HO cases are simple, and for most of them we again use Lemma C.7 by applying it to the antecedents of each rule. We include the paradigmatic case where (HO Out) is the last rule applied in the derivation of $P \xrightarrow{\lambda} P'$. We have

$$P \xrightarrow{\mathsf{out}(c_q \,:\, \rho, c_m) n \diamond [\![\, \Gamma_n \parallel c_n \parallel R \,]\!]} P'$$

because

$$P \xrightarrow{\mathsf{out}(c_q\,:\,\rho, c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m \rangle\!\rangle P_1\,.$$

So

$$P' = (\boldsymbol{\nu}\tilde{p})(P_m \mid n[\![\,\Gamma_n \parallel c_n \parallel P_1 \mid R\,]\!])\,.$$

By Lemma C.7 we have

$$Q \xrightarrow{\mathsf{out}(c_q\,:\,\rho, c_m)n} O$$

and $O \equiv (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m \rangle\!\rangle P_1$.

Now we need to consider each possible rule used to derive that equivalence. The cases of the basic rules are the most interesting ones. We show the case where (Struct ConcrNLeft) was used and $O = \langle\!\langle (\boldsymbol{\nu}\tilde{p})P_m \rangle\!\rangle\, P_1$. By (HO Out), we get

$$Q \xrightarrow{\mathsf{out}(c_q\,:\,\rho, c_m)n\diamond[\![\,\Gamma_n \parallel c_n \parallel R\,]\!]} Q'$$

where

$$Q' = (\boldsymbol{\nu}\tilde{p})P_m \mid n[\![\,\Gamma_n \parallel c_n \parallel P_1 \mid R\,]\!]\,.$$

Since $\tilde{p} \cap (\{n\} \cup \mathrm{fn}(P_1) \cup \mathrm{fn}(R)) = \varnothing$, the result follows. $\square$

In the following we will use $\xrightarrow{\lambda}\equiv$ and $\xRightarrow{\lambda}\equiv$ as shorthand notation for the composition of $\xrightarrow{\lambda}$ and $\xRightarrow{\lambda}$ (respectively) with $\equiv$.

Lemma E.2 (Higher-order transitions of processes in parallel).

(1) If $P \xrightarrow{\lambda} P'$ and $\lambda \notin \{\ \langle - \rangle^{\uparrow c} \diamond m[\![\,\Gamma_m \parallel c_m \parallel Q\,]\!] \mid Q',\ \ \mathsf{out}(c_q : \rho, c_m)n \diamond [\![\,\Gamma_n \parallel c_n \parallel Q\,]\!],$ $\mathsf{outC}(c_q : \tilde{\varphi}, c_m)n \diamond [\![\,\Gamma_n \parallel c_n \parallel Q\,]\!]\ \}$, then $P \mid R \xrightarrow{\lambda}\equiv P' \mid R$ for all processes R.

(2) If $P \xRightarrow{\hat{\lambda}} P'$ and $\lambda \notin \{\ \langle - \rangle^{\uparrow c} \diamond m[\![\,\Gamma_m \parallel c_m \parallel Q\,]\!] \mid Q',\ \ \mathsf{out}(c_q : \rho, c_m)n \diamond [\![\,\Gamma_n \parallel c_n \parallel Q\,]\!],$ $\mathsf{outC}(c_q : \tilde{\varphi}, c_m)n \diamond [\![\,\Gamma_n \parallel c_n \parallel Q\,]\!]\ \}$, then $P \mid R \xRightarrow{\hat{\lambda}}\equiv P' \mid R$ for all processes R.

(3) If $P \xrightarrow{\langle - \rangle^{\uparrow c}\diamond m[\![\,\Gamma_m \parallel c_m \parallel R\ \mid\ Q\,]\!]\ \mid\ Q'} P'$, then $P \mid R \xrightarrow{\langle - \rangle^{\uparrow c}\diamond m[\![\,\Gamma_m \parallel c_m \parallel Q\,]\!]\ \mid\ Q'}\equiv P'$.

(4) If $P \xrightarrow{\mathsf{out}(c_q\,:\,\rho, c_m)n\diamond[\![\,\Gamma_n \parallel c_n \parallel R\ \mid\ Q\,]\!]} P'$, then $P \mid R \xrightarrow{\mathsf{out}(c_q\,:\,\rho, c_m)n\diamond[\![\,\Gamma_n \parallel c_n \parallel Q\,]\!]}\equiv P'$.

(5) If $P \xrightarrow{\mathsf{outC}(c_q\,:\,\tilde{\varphi}, c_m)n\diamond[\![\,\Gamma_n \parallel c_n \parallel R\ \mid\ Q\,]\!]} P'$, then $P \mid R \xrightarrow{\mathsf{outC}(c_q\,:\,\tilde{\varphi}, c_m)n\diamond[\![\,\Gamma_n \parallel c_n \parallel Q\,]\!]}\equiv P'$.

(6) If $Q \xrightarrow{(\tilde{M})^{\star}} Q' \equiv (\boldsymbol{\nu}\tilde{n})(Q_1\{\tilde{x} := \tilde{M}\} \mid Q_2)$ and $P \xrightarrow{\langle - \rangle^{\star}\diamond R} P'$ where $R \equiv (\boldsymbol{\nu}\tilde{n})(Q_1 \mid Q_2)$, then $P \mid Q \xrightarrow{\tau}\equiv P'$.

(7) If $Q \xrightarrow{\mathsf{get}(\tilde{M}, c_n, c_m)} Q' \equiv (\boldsymbol{\nu}\tilde{n})(m[\![\,\Gamma_m \parallel c_m \parallel Q_1\{\tilde{x} := \tilde{M}\} \mid Q_2\,]\!] \mid Q_3)$ and $P \xrightarrow{\langle - \rangle^{\downarrow c_m}\diamond R} P'$ where $R \equiv (\boldsymbol{\nu}\tilde{n})(m[\![\,\Gamma_m \parallel c_m \parallel Q_1 \mid Q_2\,]\!] \mid Q_3)$, then $P \mid Q \xrightarrow{\mathsf{pre\text{-}comm}(c_n)}\equiv P'$.

(8) If $Q \xrightarrow{(\tilde{M})^{\downarrow c_m}} Q' \equiv (\boldsymbol{\nu}\tilde{n})(Q_1\{\tilde{x} := \tilde{M}\} \mid Q_2)$ and $P \xrightarrow{\mathsf{put}(c_n, c_m)\diamond R} P'$ where $R \equiv (\boldsymbol{\nu}\tilde{n})(Q_1 \mid Q_2)$, then $P \mid Q \xrightarrow{\mathsf{pre\text{-}comm}(c_n)}\equiv P'$.

(9) If $P \xrightarrow{\mathsf{in}(c_n\,:\,\rho, c_m)n\diamond[\![\,\Gamma_n \parallel c_n \parallel Q\,]\!]} P'$ and $R \xrightarrow{[\![\,\Gamma_n \parallel c_n \parallel \overline{\mathsf{in}}\,n\,]\!]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle Q \rangle\!\rangle R'$, then $P \mid R \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p})(P' \mid R')$.

(10) If $P \xrightarrow{\mathsf{in}(c_n\,:\,\rho, c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle Q \rangle\!\rangle P'$ and $R \xrightarrow{[\![\,\Gamma_n \parallel c_n \parallel \overline{\mathsf{in}}\,n\,]\!]\diamond Q} R'$, then $P \mid R \xrightarrow{\tau}\equiv (\boldsymbol{\nu}\tilde{p})(P' \mid R')$.

(11) If $P \xrightarrow{\mathsf{inC}(c_n\,:\,\tilde{\varphi},c_m)n\diamond[\,\Gamma_n \,\|\, c_n \,\|\, Q\,]} P'$ and $R \xrightarrow{[\,\Gamma_n \,\|\, c_n \,\|\, \overline{\mathsf{inC}}(c_m\,:\,\tilde{\varphi})n\,]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle Q\rangle\!\rangle R'$, then $P \mid R \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p})(P' \mid R')$.

(12) If $P \xrightarrow{\mathsf{inC}(c_n\,:\,\tilde{\varphi},c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle Q\rangle\!\rangle P'$ and $R \xrightarrow{[\,\Gamma_n \,\|\, c_n \,\|\, \overline{\mathsf{inC}}(c_m\,:\,\tilde{\varphi})n\,]\diamond Q} R'$, then $P \mid R \xrightarrow{\tau}\equiv (\boldsymbol{\nu}\tilde{p})(P' \mid R')$.

Proof. In the proofs of all cases with higher-order labels, we derive the transitions of processes and their shapes by inspection of the transition rules.

(1) If λ is a first-order label, this transition follows from rule (LTS PAR). Otherwise we will just consider the paradigmatic case where $\lambda = \mathsf{inC}(c_n : \tilde{\varphi}, c_m)n \diamond [\,\Gamma_n \,\|\, c_n \,\|\, Q\,]$. Then

$$P \xrightarrow{\mathsf{inC}(c_n\,:\,\tilde{\varphi},c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m\rangle\!\rangle P_1$$

and $P' \equiv (\boldsymbol{\nu}\tilde{p})(n[\,\Gamma_n \,\|\, c_n \,\|\, P_m \mid Q\,] \mid P_1)$. By the parallel composition rule (LTS PAR) we get

$$P \mid R \xrightarrow{\mathsf{inC}(c_n\,:\,\tilde{\varphi},c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m\rangle\!\rangle P_1 \mid R\,.$$

So we can conclude $P \mid R \xrightarrow{\lambda}\equiv P' \mid R$ by rule (HO INC).

(2) $P \overset{\hat{\lambda}}{\Longrightarrow} P'$ means either $P \Longrightarrow P'$ if $\lambda = \tau$ or $P \Longrightarrow Q \xrightarrow{\lambda} Q' \Longrightarrow P'$ for some Q, Q'. In the first case (2) follows from rule (LTS PAR); in the second case it follows from (1), Lemma E.1, and rule (LTS PAR).

(5) (The proofs of (3) and (4) are similar.)
If

$$P \xrightarrow{\mathsf{outC}(c_q\,:\,\tilde{\varphi},c_m)n\diamond[\,\Gamma_n \,\|\, c_n \,\|\, R \,\mid\, Q\,]} P'\,,$$

then

$$P \xrightarrow{\mathsf{outC}(c_q\,:\,\tilde{\varphi},c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m\rangle\!\rangle P_1$$

and

$$P' \equiv (\boldsymbol{\nu}\tilde{p})(P_m \mid n[\,\Gamma_n \,\|\, c_n \,\|\, P_1 \mid R \mid Q\,])\,.$$

By rule (LTS PAR) we get

$$P \mid R \xrightarrow{\mathsf{outC}(c_q\,:\,\tilde{\varphi},c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m\rangle\!\rangle P_1 \mid R\,.$$

So we can conclude

$$P \mid R \xrightarrow{\mathsf{outC}(c_q\,:\,\tilde{\varphi},c_m)n\diamond[\,\Gamma_n \,\|\, c_n \,\|\, Q\,]}\equiv P'$$

by rule (HO OUTC).

(7) (The proofs of (6) and (8) are similar.)
If

$$P \xrightarrow{\langle-\rangle^{\downarrow c_m}\diamond R} P'\,,$$

then

$$P \xrightarrow{\langle-\rangle^{\downarrow c_m}} (\boldsymbol{\nu}\tilde{q})\langle\!\langle\tilde{M}\rangle\!\rangle P_1$$

and

$$P' \equiv (\boldsymbol{\nu}\tilde{q})(P_1 \mid R\{\tilde{x} := \tilde{M}\}) \equiv (\boldsymbol{\nu}\tilde{q})(P_1 \mid Q'),$$

since $\mathrm{fn}(\tilde{M}) \cap \tilde{n} = \varnothing$. Finally,

$$P \mid Q \xrightarrow{\mathsf{pre\text{-}comm}(c_n)} (\boldsymbol{\nu}\tilde{q})(P_1 \mid Q') \equiv P'.$$

(11) (The proofs of (9), (10) and (12) are similar.)
If

$$P \xrightarrow{\mathsf{inC}(c_n\,:\,\tilde{\varphi},c_m)n\diamond[\![\Gamma_n \,\|\, c_n \,\|\, Q]\!]} P',$$

then

$$P \xrightarrow{\mathsf{inC}(c_n\,:\,\tilde{\varphi},c_m)n} (\boldsymbol{\nu}\tilde{q})\langle\!\langle P_m\rangle\!\rangle P_1$$

and

$$P' \equiv (\boldsymbol{\nu}\tilde{q})(n[\![\Gamma_n \,\|\, c_n \,\|\, P_m \mid Q]\!] \mid P_1).$$

We then get $P \mid R \xrightarrow{\tau}\equiv (\boldsymbol{\nu}\tilde{p})(P' \mid R')$ by rule (LTS τ-ENTER). □

Lemma E.3.

(1) If $!P \xrightarrow{\lambda} Q$, then λ is different from τ and $P \xrightarrow{\lambda} P'$ for some P'.

(2) If $P \xrightarrow{\lambda} Q$ and $q \notin \mathrm{fn}(\lambda)$, then $(\boldsymbol{\nu}q)P \xrightarrow{\lambda} (\boldsymbol{\nu}q)Q$.

Proof.

(1) The proof is immediate if we recall that P must be prefixed.

(2) When λ is first order the proof is immediate by rule (LTS RES). For higher-order labels, we consider the case $\lambda = \mathsf{outC}(c_q : \tilde{\varphi}, c_m)n \diamond [\![\Gamma_n \,\|\, c_n \,\|\, R]\!]$ only, the other cases being similar. In this case

$$P \xrightarrow{\mathsf{outC}(c_q\,:\,\tilde{\varphi},c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m\rangle\!\rangle P_1$$

and

$$Q = (\boldsymbol{\nu}\tilde{p})(n[\![\Gamma_n \,\|\, c_n \,\|\, P_1 \mid R]\!] \mid P_m).$$

By rule (LTS RES) we get

$$(\boldsymbol{\nu}q)P \xrightarrow{\mathsf{outC}(c_q\,:\,\tilde{\varphi},c_m)n} (\boldsymbol{\nu}q,\tilde{p})\langle\!\langle P_m\rangle\!\rangle P_1,$$

so we can conclude by rule (HO OUT). □

Lemma E.4. Full bisimilarity is an equivalence relation.

Proof. The proof is standard apart from the condition on types, which only matters for reflexivity. We require that P well formed and $P \xrightarrow{\lambda} P'$ imply that P' is well formed too. If λ is a first-order label, the proof by cases on λ is standard. If λ is a higher-order label, it holds by definition. Then we get that $\{(P\,;P) \mid P \text{ well-formed}\}$ is a bisimulation. □

Theorem 4.10. Full bisimilarity is a congruence.

Proof. The proof is organised in the following three steps:

A Full bisimilarity is preserved by input prefixes and by the capabilities and co-capabilities that exchange communication ports.

B Bisimilarity is preserved by capability and output prefixes and by parallel composition, ambient construction and restriction.

C Bisimilarity is preserved by replication.

Step A For input prefixes, assuming $P \approx_c Q$, we need to show that $(x : \tilde{\varphi})^\eta.P\mathsf{s} \approx (x : \tilde{\varphi})^\eta.Q\mathsf{s}$ for all closing substitutions s. By definition, we have $(x : \tilde{\varphi})^\eta.P\mathsf{s} = (x : \tilde{\varphi})^\eta.(P\mathsf{s})$ (with s capture free). The only transitions from $(x : \tilde{\varphi})^\eta.(P\mathsf{s})$ are of the form

$$(x : \tilde{\varphi})^\eta.(P\mathsf{s}) \xrightarrow{(\tilde{M})^\eta} P\mathsf{s}\{\tilde{x} := \tilde{M}\}$$

for a message $\tilde{M}$ such that $\vdash \tilde{M} : \tilde{\varphi}$. Since we also have

$$(x : \tilde{\varphi})^\eta.(Q\mathsf{s}) \xrightarrow{(\tilde{M})^\eta} Q\mathsf{s}\{\tilde{x} := \tilde{M}\},$$

it remains to prove that $P\mathsf{s}\{\tilde{x} := \tilde{M}\} \approx Q\mathsf{s}\{\tilde{x} := \tilde{M}\}$. But this follows directly from the assumption $P \approx_c Q$, since it implies $P\mathsf{s}' \approx Q\mathsf{s}'$ where $\mathsf{s}' = \mathsf{s}\{\tilde{x} := \tilde{M}\}$.

The proof is similar for the binding capabilities (and co-capabilities). For example, we need to show that $\mathsf{inC}(v{:}\tilde{\varphi})n.P\mathsf{s} \approx \mathsf{inC}(v{:}\tilde{\varphi})n.Q\mathsf{s}$, assuming that $P \approx_c Q$. Again, by definition, we have that $\mathsf{inC}(v{:}\tilde{\varphi})n.P\mathsf{s} = \mathsf{inC}(v{:}\tilde{\varphi})n.(P\mathsf{s})$ (with s capture free). As in the previous case, $\mathsf{inC}(v{:}\tilde{\varphi})n.(P\mathsf{s})$ can only move with a transition of the form

$$\mathsf{inC}(v{:}\tilde{\varphi})n.(P\mathsf{s}) \xrightarrow{\mathsf{inC}(c_n\,:\,\tilde{\varphi})n} P\mathsf{s}\{\uparrow c_n/\uparrow v\}.$$

Then

$$\mathsf{inC}(v{:}\tilde{\varphi})n.(Q\mathsf{s}) \xrightarrow{\mathsf{inC}(c_n\,:\,\tilde{\varphi})n} Q\mathsf{s}\{\uparrow c_n/\uparrow v\}.$$

We need to prove that $P\mathsf{s}\{\uparrow c_n/\uparrow v\} \approx Q\mathsf{s}\{\uparrow c_n/\uparrow v\}$. But this follows directly from the assumption $P \approx_c Q$ since it implies $P\mathsf{s}' \approx Q\mathsf{s}'$ where $\mathsf{s}' = \mathsf{s}\{\uparrow c_n/\uparrow v\}$.

Step B Since we know that $\approx_c$ is preserved by input prefixes, we can consider $\approx$. We define $\mathscr{B}$ as the *contextual* closure of $\approx$ with respect to capability and output prefixes and by parallel composition, ambient construction and restriction, that is, as the least symmetric relation such that:

(1) $\approx \subseteq \mathscr{B}$.

(2) $P \mathscr{B} Q$ implies $C.P \mathscr{B} C.Q$.

(3) $P \mathscr{B} Q$ implies $\langle \tilde{M} \rangle^\eta.P \mathscr{B} \langle \tilde{M} \rangle^\eta.Q$.

(4) $P \mathscr{B} Q$ implies $P \mid R \mathscr{B} Q \mid R$ and $R \mid P \mathscr{B} R \mid Q$.

(5) $P \mathscr{B} Q$ implies $n[\![\, \Gamma \parallel c \parallel P \,]\!] \mathscr{B} n[\![\, \Gamma \parallel c \parallel Q \,]\!]$.

(6) $P \mathscr{B} Q$ implies $(\boldsymbol{\nu} n)P \mathscr{B} (\boldsymbol{\nu} n)Q$.

It is clearly enough to show that $\mathscr{B}$ is a bisimulation up to $\equiv$, since this implies $\mathscr{B} \subseteq \approx$, and we conclude $\mathscr{B} = \approx$, which proves that $\approx$ (and then $\approx_c$) is preserved by the listed process constructors. The proof is by induction on the definition of $\mathscr{B}$ using Lemmas E.2 and E.3.

(1) This condition follows by definition.

(2) Note that if $C.P \xrightarrow{\lambda} P'$, then $\lambda = C \in \{\mathsf{in}\ \alpha, \mathsf{out}\ \alpha, \overline{\mathsf{in}}, \overline{\mathsf{out}}\}$ and $P' \equiv P$. Since we also have $C.Q \xrightarrow{C} Q$ for $C \in \{\mathsf{in}\ \alpha, \mathsf{out}\ \alpha, \overline{\mathsf{in}}, \overline{\mathsf{out}}\}$, we are done.

(3) This proof is similar to (2).

(4) We need to consider many different subcases:

— If $P \mid R \xrightarrow{\lambda} P \mid R'$ because $R \xrightarrow{\lambda} R'$ and

$$\begin{aligned}\lambda \notin \{ & \langle - \rangle^{\uparrow c} \diamond m[\![\, \Gamma_m \parallel c_m \parallel Q \,]\!] \mid Q', \\ & \mathsf{out}(c_q : \rho, c_m)n \diamond [\![\, \Gamma_n \parallel c_n \parallel Q \,]\!], \\ & \mathsf{outC}(c_q : \tilde{\varphi}, c_m)n \diamond [\![\, \Gamma_n \parallel c_n \parallel Q \,]\!] \, \}.\end{aligned}$$

The proof follows trivially by Lemma E.2(1) and contextuality of $\mathscr{B}$.

— Let $P \mid R \xrightarrow{\lambda} P' \mid R$ because $P \xrightarrow{\lambda} P'$ and

$$\begin{aligned}\lambda \notin \{ & \langle - \rangle^{\uparrow c} \diamond m[\![\, \Gamma_m \parallel c_m \parallel Q \,]\!] \mid Q', \\ & \mathsf{out}(c_q : \rho, c_m)n \diamond [\![\, \Gamma_n \parallel c_n \parallel Q \,]\!], \\ & \mathsf{outC}(c_q : \tilde{\varphi}, c_m)n \diamond [\![\, \Gamma_n \parallel c_n \parallel Q \,]\!] \, \}.\end{aligned}$$

Then, by induction, $Q \stackrel{\hat{\lambda}}{\Longrightarrow} Q'$ for some Q' such that $P' \mathscr{B} Q'$. By Lemma E.2(2), we get $Q \mid R \stackrel{\hat{\lambda}}{\Longrightarrow} \equiv Q' \mid R$, and then $P' \mid R \mathscr{B} Q' \mid R$ by the contextuality of $\mathscr{B}$.

— Let

$$P \mid R \xrightarrow{\langle - \rangle^{\uparrow c} \diamond m[\![\, \Gamma_m \parallel c_m \parallel S_1 \,]\!] \mid S_2} R'$$

because

$$R \xrightarrow{\langle - \rangle^{\uparrow c}} (\boldsymbol{\nu} \tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle R_1 .$$

Then

$$R' = (\boldsymbol{\nu} \tilde{p})(m[\![\, \Gamma_m \parallel c_m \parallel P \mid R_1 \mid S_1 \,]\!] \mid S_2\{\tilde{x} := \tilde{M}\}).$$

By rule (LTS PAR), we have

$$Q \mid R \xrightarrow{\langle - \rangle^{\uparrow c}} (\boldsymbol{\nu} \tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle Q \mid R_1$$

and then, by rule (HO SEND$^{\uparrow}$), we get

$$Q \mid R \xrightarrow{\langle - \rangle^{\uparrow c} \diamond m[\![\, \Gamma_m \parallel c_m \parallel S_1 \,]\!] \mid S_2} (\boldsymbol{\nu} \tilde{p})(m[\![\, \Gamma_m \parallel c_m \parallel Q \mid R_1 \mid S_1 \,]\!] \mid S_2\{\tilde{x} := \tilde{M}\}),$$

so we are done by the contextuality of $\mathscr{B}$.

— The proofs in cases

$$P \mid R \xrightarrow{\xi \diamond [\![\, \Gamma_n \parallel c_n \parallel S \,]\!]} R'$$

with

$$\xi \in \{\mathsf{out}(c_q : \rho, c_m)n, \mathsf{outC}(c_q : \tilde{\varphi}, c_m)n\}$$

because $R \xrightarrow{\xi} (\boldsymbol{\nu} \tilde{p})\langle\!\langle R_1 \rangle\!\rangle R_2$ are similar to the previous case.

— Let

$$P \mid R \xrightarrow{\langle - \rangle^{\uparrow c} \diamond m[\![\, \Gamma_m \parallel c_m \parallel S_1 \,]\!] \mid S_2} P'$$

because

$$P \xrightarrow{\langle - \rangle^{\uparrow c}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P_1 \, .$$

Then

$$P' = (\boldsymbol{\nu}\tilde{p})(m[\![\, \Gamma_m \parallel c_m \parallel P_1 \mid R \mid S_1 \,]\!] \mid S_2\{\tilde{x} := \tilde{M}\}) \, .$$

By rule (HO Send$^{\uparrow}$) we get

$$P \xrightarrow{\langle - \rangle^{\uparrow c} \diamond m[\![\, \Gamma_m \parallel c_m \parallel R \mid S_1 \,]\!] \mid S_2} P' \, .$$

Then, by induction,

$$Q \Longrightarrow Q_1 \xrightarrow{\langle - \rangle^{\uparrow c} \diamond m[\![\, \Gamma_m \parallel c_m \parallel R \mid S_1 \,]\!] \mid S_2} Q_2 \Longrightarrow Q'$$

for some Q_1, Q_2, and Q' such that $P' \mathscr{B} Q'$. By rule (lts Par), we have $Q \mid R \Longrightarrow Q_1 \mid R$, and by Lemma E.2(3), we have

$$Q_1 \mid R \xrightarrow{\langle - \rangle^{\uparrow c} \diamond m[\![\, \Gamma_m \parallel c_m \parallel S_1 \,]\!] \mid S_2} \equiv Q_2 \, ,$$

so we are done.

— The proofs in cases

$$P \mid R \xrightarrow{\xi \diamond [\![\, \Gamma_n \parallel c_n \parallel S \,]\!]} P'$$

with $\xi \in \{\mathsf{out}(c_q : \rho, c_m)n, \mathsf{outC}(c_q : \tilde{\varphi}, c_m)n\}$ because $P \xrightarrow{\xi} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2$ are similar to the previous case.

— If

$$P \mid R \xrightarrow{\mathsf{pre\text{-}comm}(c_n)} P'$$

because

$$P \xrightarrow{\langle - \rangle^{\downarrow c_m}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P_1 \quad \text{ and } \quad R \xrightarrow{\mathsf{get}(\tilde{M}, c_n, c_m)} R_1 \, ,$$

then $\mathrm{fn}(R) \cap \tilde{p} = \varnothing$ and $P' \equiv (\boldsymbol{\nu}\tilde{p})(P_1 \mid R_1)$. By Lemma C.1(3) we have

$$R \equiv (\boldsymbol{\nu}\tilde{q})(m[\![\, \Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi})^{\uparrow c_n}.S_1 \mid S_2 \,]\!] \mid S_3)$$

and

$$R_1 \equiv (\boldsymbol{\nu}\tilde{q})(m[\![\, \Gamma_m \parallel c_m \parallel S_1\{\tilde{x} := \tilde{M}\} \mid S_2 \,]\!] \mid S_3) \, .$$

We can assume that $(\mathrm{fv}(S_2) \cup \mathrm{fv}(S_3)) \cap \tilde{x} = \varnothing$. Let

$$R_2 \equiv (\boldsymbol{\nu}\tilde{q})(m[\![\, \Gamma_m \parallel c_m \parallel S_1 \mid S_2 \,]\!] \mid S_3) \, .$$

By (HO Put) and induction, we have

$$P \xrightarrow{\langle - \rangle^{\downarrow c_m} \diamond R_2} P'$$

and

$$Q \Longrightarrow Q_1 \xrightarrow{\langle - \rangle^{\downarrow c_m} \diamond R_2} Q_2 \Longrightarrow Q'$$

for some Q_1, Q_2 and Q' such that $P' \mathscr{B} Q'$. By (LTS PAR), we have $Q \mid R \Longrightarrow Q_1 \mid R$ and by Lemma E.2 (7), we have

$$Q_1 \mid R \xrightarrow{\mathsf{pre\text{-}comm}(c_n)} \equiv Q_2 \Longrightarrow Q' \, .$$

— The proofs in cases $P \mid R \xrightarrow{\mathsf{pre\text{-}comm}(c_n)} P'$ because

– $P \xrightarrow{\mathsf{get}(\tilde{M},c_n,c_m)} P_1$ and $R \xrightarrow{\langle - \rangle^{\downarrow c_m}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle R_1$

– $P \xrightarrow{\mathsf{put}(c_n,c_m)} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P_1$ and $R \xrightarrow{(\tilde{M})^{\downarrow c_m}} R_1$

– $P \xrightarrow{(\tilde{M})^{\downarrow c_m}} P_1$ and $R \xrightarrow{\mathsf{put}(c_n,c_m)} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle R_1$

are similar to the previous case.

— The proofs in cases $P \mid R \xrightarrow{\tau} P'$ because

– $P \xrightarrow{\langle - \rangle^{\star}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P_1$ and $R \xrightarrow{(\tilde{M})^{\star}} R_1$

– $P \xrightarrow{(\tilde{M})^{\star}} P_1$ and $R \xrightarrow{\langle - \rangle^{\star}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle R_1$

are simpler than the previous cases.

— If

$$P \mid R \xrightarrow{\mathsf{pre\text{-}exitC}(c\,:\,\tilde{\varphi},c_m)} P' \mid R'$$

because

$$P \xrightarrow{\mathsf{popC}(c\,:\,\tilde{\varphi},c_m)} P' \quad \text{and} \quad R \xrightarrow{\overline{\mathsf{outC}}(c_m\,:\,\tilde{\varphi})} R' \, ,$$

then by the induction hypothesis

$$Q \Longrightarrow Q_1 \xrightarrow{\mathsf{popC}(c\,:\,\tilde{\varphi},c_m)} Q_2 \Longrightarrow Q'$$

and $P' \mathscr{B} Q'$. By rule (LTS PRE-EXITC) we get

$$Q \mid R \Longrightarrow Q_1 \mid R \xrightarrow{\mathsf{pre\text{-}exitC}(c\,:\,\tilde{\varphi},c_m)} Q_2 \mid R' \Longrightarrow Q' \mid R'$$

and the case follows by contextuality of $\mathscr{B}$.

— The proof for the case $P \mid R \xrightarrow{\mathsf{pre\text{-}exitC}(c\,:\,\tilde{\varphi},c_m)} P' \mid R'$ because $P \xrightarrow{\overline{\mathsf{outC}}(c_m\,:\,\tilde{\varphi})} P'$ and $R \xrightarrow{\mathsf{popC}(c\,:\,\tilde{\varphi},c_m)} R'$ is analogous to the previous case.

— The proofs for the cases $P \mid R \xrightarrow{\mathsf{pre\text{-}exit}(c_p\,:\,\rho,c_m)} P' \mid R'$ are similar to the previous case.

— If $P \mid R \xrightarrow{\tau} P'$ because

$$P \xrightarrow{\mathsf{in}(c_n\,:\,\rho,c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \text{and} \quad R \xrightarrow{[\,\Gamma_n \,\|\, c_n \,\|\, \overline{\mathsf{in}}\, n\,]} (\boldsymbol{\nu}\tilde{q})\langle\!\langle R_1 \rangle\!\rangle R_2 \, ,$$

then

$$P' \equiv (\boldsymbol{\nu}\tilde{p},\tilde{q})(n[\,\Gamma_n \,\|\, c_n \,\|\, P_1 \mid R_1\,] \mid P_2 \mid R_2) \, .$$

By rule (HO SEND$^{\uparrow}$) we get

$$P \xrightarrow{\mathsf{in}(c_n\,:\,\rho,c_m)n\diamond[\,\Gamma_n \,\|\, c_n \,\|\, R_1\,]} (\boldsymbol{\nu}\tilde{p})(n[\,\Gamma_n \,\|\, c_n \,\|\, P_1 \mid R_1\,] \mid P_2) \, .$$

Then, by induction,

$$Q \Longrightarrow Q_1 \xrightarrow{\mathsf{in}(c_n\,:\,\rho,c_m)n\diamond[\,\Gamma_n \parallel c_n \parallel R_1\,]} Q_2 \Longrightarrow Q'$$

for some Q_1, Q_2 and Q' such that

$$(\boldsymbol{\nu}\tilde{p})(n[\,\Gamma_n \parallel c_n \parallel P_1 \mid R_1\,] \mid P_2)\,\mathscr{B}\,Q'\,.$$

By rule (LTS PAR), we have $Q \mid R \Longrightarrow Q_1 \mid R$, and by Lemma E.2(9), we have $Q_1 \mid R \Longrightarrow\equiv (\boldsymbol{\nu}\tilde{q})(Q_2 \mid R_2)$, and this concludes the proof by contextuality of $\mathscr{B}$.

— The proofs in the cases $P \mid R \xrightarrow{\tau} P'$ because

– $P \xrightarrow{[\,\Gamma_n \parallel c_n \parallel \overline{\mathsf{in}}\,n\,]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1\rangle\!\rangle P_2$ and $R \xrightarrow{\mathsf{in}(c_n\,:\,\rho,c_m)n} (\boldsymbol{\nu}\tilde{q})\langle\!\langle R_1\rangle\!\rangle R_2$

– $P \xrightarrow{\mathsf{inC}(c_n\,:\,\tilde{\varphi},c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1\rangle\!\rangle P_2$ and $R \xrightarrow{[\,\Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(c_m\,:\,\tilde{\varphi})n\,]} (\boldsymbol{\nu}\tilde{q})\langle\!\langle R_1\rangle\!\rangle R_2$

– $P \xrightarrow{[\,\Gamma_n \parallel c_n \parallel \overline{\mathsf{inC}}(c_m\,:\,\tilde{\varphi})n\,]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1\rangle\!\rangle P_2$ and $R \xrightarrow{\mathsf{inC}(c_n\,:\,\tilde{\varphi},c_m)n} (\boldsymbol{\nu}\tilde{q})\langle\!\langle R_1\rangle\!\rangle R_2$

are similar to the previous case.

(5) This condition also requires us to examine different cases:

— The case $n[\,\Gamma \parallel c \parallel P\,] \xrightarrow{\tau} n[\,\Gamma \parallel c \parallel P'\,]$ because $P \xrightarrow{\tau} P'$ is trivial.

— Let

$$m[\,\Gamma_m \parallel c_m \parallel P\,] \xrightarrow{\mathsf{get}(\tilde{M},c_n,c_m)} m[\,\Gamma_m \parallel c_m \parallel P'\,]$$

because

$$P \xrightarrow{(\tilde{M})^{\uparrow c_n}} P'\,.$$

By induction,

$$Q \Longrightarrow Q_1 \xrightarrow{\langle\tilde{M}\rangle^{\uparrow c_n}} Q_2 \Longrightarrow Q'$$

for some Q_1, Q_2 and Q' such that $P'\,\mathscr{B}\,Q'$. Note that $m[\,\Gamma_m \parallel c_m \parallel Q\,]$ is a well-formed process since $m[\,\Gamma_m \parallel c_m \parallel P\,]$ is well formed and $P\,\mathscr{B}\,Q$. We have

$$m[\Gamma_m \parallel c_m \parallel Q] \Longrightarrow m[\Gamma_m \parallel c_m \parallel Q_1]$$

and

$$m[\Gamma_m \parallel c_m \parallel Q_2] \Longrightarrow m[\Gamma_m \parallel c_m \parallel Q']$$

by rule (LTS AMB). Moreover,

$$m[\,\Gamma_m \parallel c_m \parallel Q_1\,] \xrightarrow{\mathsf{get}(\tilde{M},c_n,c_m)} m[\,\Gamma_m \parallel c_m \parallel Q_2\,]$$

by rule (LTS GET), and this concludes the proof.

— The proofs in cases:

– $n[\,\Gamma_n \parallel c_n \parallel P\,] \xrightarrow{\tau} n[\,\Gamma_n \parallel c_n \parallel P'\,]$ because $P \xrightarrow{\mathsf{pre\text{-}comm}(c_n)} P'$

– $p[\,\Gamma_p \parallel c_p \parallel P\,] \xrightarrow{\tau} p[\,\Gamma_p \parallel c_p \parallel P'\,]$ because $P \xrightarrow{\mathsf{pre\text{-}exit}(c_p\,:\,\rho,c_m)} P'$

– $p[\,\Gamma_p \parallel c_p \parallel P\,] \xrightarrow{\tau} p[\,\Gamma_p \oplus \tilde{\varphi}^{\downarrow c_m} \parallel c_p \parallel P'\,]$ because $P \xrightarrow{\mathsf{pre\text{-}exitC}(c_p\,:\,\tilde{\varphi},c_m)} P'$

are similar to the previous case.

— Let

$$n[\,\Gamma_n \parallel c_n \parallel P\,] \xrightarrow{\mathsf{pop}(c\,:\,\rho,c_m)} (\boldsymbol{\nu}\tilde{p})(n[\,\Gamma_n \parallel c_n \parallel P_n\,] \mid P_m)$$

because

$$P \xrightarrow{\mathsf{out}(c\,:\,\rho,c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m\rangle\!\rangle P_n\,.$$

By rule (HO Out), we get

$$P \xrightarrow{\mathsf{out}(c_q\,:\,\rho,c_m)n\diamond[\,\Gamma_n \parallel c_n \parallel \mathbf{0}]} (\boldsymbol{\nu}\tilde{p})(n[\,\Gamma_n \parallel c_n \parallel P_n \mid \mathbf{0}\,] \mid P_m)\,.$$

By induction,

$$Q \Longrightarrow Q_1 \xrightarrow{\mathsf{out}(c_q\,:\,\rho,c_m)n\diamond[\,\Gamma_n \parallel c_n \parallel \mathbf{0}]} Q_2 \Longrightarrow Q'$$

for some Q_1, Q_2 and Q' such that $(\boldsymbol{\nu}\tilde{p})(n[\,\Gamma_n \parallel c_n \parallel P_n \mid \mathbf{0}\,] \mid P_m)\,\mathscr{B}\,Q'$. We have

$$n[\,\Gamma_n \parallel c_n \parallel Q\,] \Longrightarrow n[\,\Gamma_n \parallel c_n \parallel Q_1\,]$$

and

$$n[\,\Gamma_n \parallel c_n \parallel Q_2\,] \Longrightarrow n[\,\Gamma_n \parallel c_n \parallel Q'\,]$$

by rule (LTS Amb). From

$$Q_1 \xrightarrow{\mathsf{out}(c_q\,:\,\rho,c_m)n\diamond[\,\Gamma_n \parallel c_n \parallel \mathbf{0}]} Q_2$$

we get

$$Q_1 \xrightarrow{\mathsf{out}(c\,:\,\rho,c_n)n} (\boldsymbol{\nu}\tilde{q})\langle\!\langle Q_m\rangle\!\rangle Q_n$$

and

$$Q_2 \equiv (\boldsymbol{\nu}\tilde{q})(n[\,\Gamma_n \parallel c_n \parallel Q_n \mid \mathbf{0}\,] \mid Q_m)\,.$$

By rule (LTS Pop) we get

$$n[\,\Gamma_n \parallel c_n \parallel Q_1\,] \xrightarrow{\mathsf{pop}(c\,:\,\rho,c_n)} \equiv Q_2\,,$$

and this concludes the proof.

— The proof in case

$$n[\,\Gamma_n \parallel c_n \parallel P\,] \xrightarrow{\mathsf{popC}(c\,:\,\tilde{\varphi},c_m)} (\boldsymbol{\nu}\tilde{p})(n[\,\Gamma_n \parallel c_n \parallel P_n\,] \mid P_m)$$

because

$$P \xrightarrow{\mathsf{outC}(c\,:\,\tilde{\varphi},c_m)n} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_m\rangle\!\rangle P_n$$

is similar to the previous case.

(6) We also have several cases for this condition. Let

$$(\boldsymbol{\nu}q)P \xrightarrow{\mathsf{outC}(c_q\,:\,\tilde{\varphi},c_m)n\diamond[\,\Gamma_n \parallel c_n \parallel R\,]} P'$$

because

$$(\boldsymbol{\nu}q)P \xrightarrow{\mathsf{outC}(c_q\,:\,\tilde{\varphi},c_m)n} (\boldsymbol{\nu}q,\tilde{p})\langle\!\langle P_m\rangle\!\rangle P_n\,.$$

Then $P' \equiv (\boldsymbol{\nu}q)P''$ and

$$P \xrightarrow{\mathsf{outC}(c_q\,:\,\tilde{\varphi},c_m)n\diamond[\,\Gamma_n \parallel c_n \parallel R\,]} P''$$

where

$$P'' \equiv (\boldsymbol{\nu}\tilde{p})(P_m \mid n[\![\, \Gamma_n \parallel c_n \parallel P_n \mid R \,]\!])\,.$$

By induction,

$$Q \Longrightarrow Q_1 \xrightarrow{\mathsf{outC}(c_q\,:\,\tilde{\varphi},c_m)n\diamond[\![\,\Gamma_n \parallel c_n \parallel R\,]\!]} Q_2 \Longrightarrow Q''$$

and $P'' \mathscr{B}\, Q''$ for some Q_1, Q_2, Q''. As q is bound, we can assume

$$q \notin \mathrm{fn}(\mathsf{outC}(c_q : \tilde{\varphi}, c_m)n \diamond [\![\, \Gamma_n \parallel c_n \parallel R \,]\!])\,.$$

By Lemma E.3(2),

$$(\boldsymbol{\nu} q)Q \Longrightarrow (\boldsymbol{\nu} q)Q_1 \xrightarrow{\mathsf{outC}(c_q\,:\,\tilde{\varphi},c_m)n\diamond[\![\,\Gamma_n \parallel c_n \parallel R\,]\!]} (\boldsymbol{\nu} q)Q_2 \Longrightarrow (\boldsymbol{\nu} q)Q''\,,$$

and $(\boldsymbol{\nu} q)Q'' \mathscr{B}\, (\boldsymbol{\nu} q)P'' \equiv P'$ by contextuality of $\mathscr{B}$.

The proofs in the other cases are similar.

Step C For the final step, we consider the relation $\mathscr{C}$ defined as $\mathscr{B}$ plus a rule for replication:

(1) $\approx \subseteq \mathscr{C}$;

(2) $P \,\mathscr{C}\, Q$ implies $C.P \,\mathscr{C}\, C.Q$;

(3) $P \,\mathscr{C}\, Q$ implies $\langle \tilde{M} \rangle^{\eta}.P \,\mathscr{C}\, \langle \tilde{M} \rangle^{\eta}.Q$;

(4) $P \,\mathscr{C}\, Q$ implies $P \mid R \,\mathscr{C}\, Q \mid R$ and $R \mid P \,\mathscr{C}\, R \mid Q$;

(5) $P \,\mathscr{C}\, Q$ implies $n[\![\, \Gamma \parallel c \parallel P \,]\!] \,\mathscr{C}\, n[\![\, \Gamma \parallel c \parallel Q \,]\!]$;

(6) $P \,\mathscr{C}\, Q$ implies $(\boldsymbol{\nu} n)P \,\mathscr{C}\, (\boldsymbol{\nu} n)Q$;

(7) $P \approx Q$ implies $!P \,\mathscr{C}\, !Q$, where P, Q are prefixed processes.

We prove that

— If $P \xrightarrow{\lambda} P'$ where $\lambda \neq \tau$, then $Q \stackrel{\lambda}{\Longrightarrow} Q'$ and $P' \approx \mathscr{C} \approx Q'$.

— If $P \xrightarrow{\tau} P'$, then $Q \Longrightarrow Q'$ and $P' \equiv \mathscr{C} \equiv Q'$.

— If $Q \xrightarrow{\lambda} Q'$ where $\lambda \neq \tau$, then $P \stackrel{\lambda}{\Longrightarrow} P'$ and $Q' \approx \mathscr{C} \approx P'$.

— If $Q \xrightarrow{\tau} Q'$, then $P \Longrightarrow P'$ and $Q' \equiv \mathscr{C} \equiv P'$.

From the above we can conclude that $\mathscr{C}$ is a bisimilarity using Exercise 2.4.64 of Sangiorgi and Walker (2002), and hence, $\mathscr{C} = \approx$.

The proofs for the first six cases are exactly as in **Step B**.

For (7), note first that if $!P \xrightarrow{\lambda} P'$, then $P \xrightarrow{\lambda} P_1$ and $\lambda \neq \tau$ by Lemma E.3(1). If λ is a first-order label, we get $P' \equiv\, !P \mid P_1$. By definition, $Q \xrightarrow{\lambda} Q_2 \Longrightarrow Q_1$ and $P_1 \approx Q_1$ for some Q_2, Q_1. By rules (LTS REPL) and (LTS PAR) $!Q \xrightarrow{\lambda} !Q \mid Q_2 \Longrightarrow !Q \mid Q_1$. Finally, note that $!P \mid P_1 \,\mathscr{C}\, !Q \mid P_1 \approx\, !Q \mid Q_1$, since in **Step B** we proved that $\approx$ is preserved by parallel composition.

If λ is a higher-order label, note that only the HO rules (HO SEND $\star$-$\Downarrow$) and (HO SEND$^{\uparrow}$) can be used. We give the proof for $\lambda = \langle - \rangle^{\uparrow c} \diamond m[\![\, \Gamma_m \parallel c_m \parallel R \,]\!] \mid S$, that is, for rule (HO SEND$^{\uparrow}$). The proof for (HO SEND $\star$-$\Downarrow$) is simpler.

We have

$$P \xrightarrow{\langle - \rangle^{\uparrow c}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P_1$$

and

$$P' = (\boldsymbol{\nu}\tilde{p})(m[\![\Gamma_m \parallel c_m \parallel P_1 \mid !P \mid R]\!] \mid S\{\tilde{x} := \tilde{M}\}).$$

By rule (HO Send$^{\uparrow}$),

$$P \xrightarrow{\langle - \rangle^{\uparrow c} \diamond m[\![\Gamma_m \parallel c_m \parallel !Q \mid R]\!] \mid S} P'',$$

where

$$P'' = (\boldsymbol{\nu}\tilde{p})(m[\![\Gamma_m \parallel c_m \parallel P_1 \mid !Q \mid R]\!] \mid S\{\tilde{x} := \tilde{M}\}).$$

Note that $P' \mathscr{C} P''$ by the definition of $\mathscr{C}$. By definition,

$$Q \xrightarrow{\langle - \rangle^{\uparrow c} \diamond m[\![\Gamma_m \parallel c_m \parallel !Q \mid R]\!] \mid S} Q' \Longrightarrow Q''$$

for some Q', Q'' such that $P'' \approx Q''$. Then, by Lemma E.2(3),

$$Q \mid !Q \xrightarrow{\langle - \rangle^{\uparrow c} \diamond m[\![\Gamma_m \parallel c_m \parallel R]\!] \mid S} \equiv Q'.$$

By Lemma E.1, we get

$$!Q \xrightarrow{\langle - \rangle^{\uparrow c} \diamond m[\![\Gamma_m \parallel c_m \parallel !Q \mid R]\!] \mid S} T$$

for some $T \equiv Q'$, and $T \Longrightarrow T'$ for some $T' \equiv Q''$. Finally, $P' \mathscr{C} P'' \approx Q'' \equiv T'$, so we are done.

Note that the above proof uses the restriction that all replicated processes are prefixed. □

Appendix F. Soundness of full bisimilarity (Theorem 4.11)

Lemma F.1. Full bisimilarity is barb preserving over closed processes.

Proof. Suppose P, Q are closed processes, $P \approx Q$ and $P \downarrow^1_{(n)}$. By Lemma 4.7, $P \downarrow^1_{(n)}$ implies

$$P \xrightarrow{[\![\Gamma \parallel c \parallel \overline{\mathsf{in}}\ n]\!] \diamond R} P'$$

for some Γ, c, R, P'. Then, since $P \approx Q$, we get

$$Q \overset{[\![\Gamma \parallel c \parallel \overline{\mathsf{in}} n]\!] \diamond R}{\Longrightarrow} Q'$$

for some Q'. In particular, there are Q_1, Q_2 such that

$$Q \Longrightarrow Q_1 \xrightarrow{[\![\Gamma \parallel c \parallel \overline{\mathsf{in}} n]\!] \diamond R} Q_2 \Longrightarrow Q'.$$

From Lemma 4.7 we deduce $Q_1 \downarrow^1_{(n)}$, and hence $Q \Downarrow^1_{(n)}$, as required. □

Theorem 4.11 (Soundness of full bisimilarity). If $P \approx_c Q$, then $P \cong Q$.

Proof. It suffices to show that $\approx_c$ is a barbed bisimulation. This follows from the fact that $\approx_c$:

1 is a congruence (This follows from Theorem 4.10.);
2 is reduction closed on closed processes (Suppose P, Q are closed processes, $P \approx_c Q$ and $P \longrightarrow P'$. By Theorem 4.6, $P \xrightarrow{\tau} P_1$ and $P_1 \equiv P'$. Since $P \approx_c Q$, there exists Q' such

that $Q \Longrightarrow Q'$ and $P' \equiv P_1 \approx_c Q'$. By Lemma E.1 and by transitivity of $\approx_c$ we are done.);

3 Is barb preserving on closed processes (This follows from Lemma F.1.). □

Acknowledgements

We thank Healfdene Goguen for helpful comments on an earlier draft. We gratefully acknowledge (in alphabetic order) Michele Bugliesi, Mario Coppo, Massimo Merro and Vladimiro Sassone for enlightening discussions on labelled transition systems for ambient calculi. The present version of this paper has been greatly improved thanks to the useful suggestions of careful referees. In particular, a mistake in the proof of Theorem 4.10 has been corrected.

References

Amtoft, T., Kfoury, A.J. and Pericas-Geertsen, S.M. (2001) What are Polymorphically-Typed Ambients? In: Sands, D. (ed.) ESOP'01. *Springer-Verlag Lecture Notes in Computer Science* **2028** 206–220.

Amtoft, T., Makholm, H. and Wells, J.B. (2004) PolyA: True Type Polymorphism for Mobile Ambients. In: Lévy, J.-J., Mayr, E.W. and Mitchell, J.C. (eds.) *TCS'04* 591–604.

Barbanera, F., Bugliesi, M., Dezani-Ciancaglini, M. and Sassone, V. (2003) A Calculus of Bounded Capacities. In: Saraswat, V.A. (ed.) ASIAN'03. *Springer-Verlag Lecture Notes in Computer Science* **2896** 205–223.

Bonelli, E., Compagnoni, A. and Gunter, E. (2005) Correspondence Assertions for Process Synchronization in Concurrent Communications. *Journal of Functional Programming* **15** (2) 219–248.

Bonelli, E., Compagnoni, A.B., Dezani-Ciancaglini, M. and Garralda, P. (2004) Boxed Ambients with Communication Interfaces. In: Fiala, J., Koubek, V. and Kratochvíl, J. (eds.) MFCS'04. *Springer-Verlag Lecture Notes in Computer Science* **3153** 119–148.

Boudol, G. (2003) A Parametric Model of Migration and Mobility, Release 1. Mikado Deliverable D1.2.1. (Available at `http://mikado.di.fc.ul.pt/repository/D1.2.1.pdf`.)

Bugliesi, M. and Castagna, G. (2002) Behavioral Typing for Safe Ambients. *Computer Languages* **28** (1) 61–99.

Bugliesi, M., Castagna, G. and Crafa, S. (2004) Access Control for Mobile Agents: The Calculus of Boxed Ambients. *ACM Transactions on Programming Languages and Systems* **26** (1) 57–124.

Bugliesi, M., Crafa, S., Merro, M. and Sassone, V. (2005) Communication and Mobility Control in Boxed Ambients. *Information and Computation* **202** (1) 39–86.

Cardelli, L., Ghelli, G. and Gordon, A.D. (2002) Types for the Ambient Calculus. *Information and Computation* **177** (2) 160–194.

Cardelli, L. and Gordon, A.D. (2000) Mobile Ambients. *Theoretical Computer Science* **240** (1) (Special Issue on Coordination) 177–213.

Castagna, G., Vitek, J. and Nardelli, F.Z. (2005) The Seal Calculus. *Information and Computation* **201** (1) 1–54.

Compagnoni, A. and Gunter, E. (2005) Types for Security in a Mobile World. In: Nicola, R.D. and Sangiorgi, D. (eds.) TGC 2005. *Springer-Verlag Lecture Notes in Computer Science* **3705** 75–97.

Coppo, M., Cozzi, F., Dezani-Ciancaglini, M., Giovannetti, E. and Pugliese, R. (2005) A Mobility Calculus with Local and Dependent Types. In: Middeldorp, A., van Oostrom, V., van Raamsdonk, F. and de Vrijer, R. (eds.) Processes, Terms and Cycles: Steps on the Road to Infinity. *Springer-Verlag Lecture Notes in Computer Science* **3838** 404–444.

Coppo, M., Dezani-Ciancaglini, M., Giovannetti, E. and Pugliese, R. (2004) Dynamic and Local Typing for Mobile Ambients. In: Lévy, J.-J., Mayr, E. W. and Mitchell, J. C. (eds.) *TCS'04*, Kluwer 583–596.

Coppo, M., Dezani-Ciancaglini, M., Giovannetti, E. and Salvo, I. (2003) M3: Mobility Types for Mobile Processes in Mobile Ambients. In: Harland, J. (ed.) CATS'03. *Electronic Notes in Theoretical Computer Science* **78**.

De Nicola, R., Ferrari, G. and Pugliese, R. (1998) Klaim: a Kernel Language for Agents Interaction and Mobility. *IEEE Transactions on Software Engineering* **24** (5) 315–330.

Dezani-Ciancaglini, M., Mostrous, D., Yoshida, N. and Drossopoulou, S. (2006) Session Types for Object-Oriented Languages. In: Thomas, D. (ed.) ECOOP'06. *Springer-Verlag Lecture Notes in Computer Science* **4067** 328–352.

Garralda, P. and Compagnoni, A. (2005) Splitting Mobility and Communication in Boxed Ambients. In: Fernández, M. and Mackie, I. (eds.) DCM 2005. *Electronic Notes in Theoretical Computer Science* **135** 105–114.

Garralda, P., Compagnoni, A. and Dezani-Ciancaglini, M. (2006) BASS: Boxed Ambients with Safe Sessions. In: Maher, M. (ed.) *PPDP'06*, ACM Press 61–72.

Giovannetti, E. (2003) Ambient Calculi with Types: a Tutorial. In: Priami, C. (ed.) Global Computing. *Springer-Verlag Lecture Notes in Computer Science* **2874** 151–191.

Goguen, H. (1995) Typed Operational Semantics. In: Dezani-Ciancaglini, M. and Plotkin, G. (eds.) TLCA'95. *Springer-Verlag Lecture Notes in Computer Science* **902** 186–200.

Goguen, H. (1999) Soundness of a Typed Operational Semantics for the Logical Framework. In: Girard, J.-Y. (ed.) TLCA'99. *Springer-Verlag Lecture Notes in Computer Science* **1581** 177–197.

Gordon, A. D. and Cardelli, L. (2003) Equational Properties of Mobile Ambients. *Mathematical Structures in Computer Science* **13** (3) 371–408.

Hennessy, M., Merro, M. and Rathke, J. (2004) Towards a Behavioural Theory of Access and Mobility Control in Distributed Systems. *Theoretical Computer Science* **322** (3) 615–669.

Hennessy, M. and Riely, J. (2002) Resource Access Control in Systems of Mobile Agents. *Information and Computation* **173** 82–120.

Hennessy, M. and Riely, J. (2003) Trust and Partial Typing in Open Systems of Mobile Agents. *Journal of Automated Reasoning* **31** (3-4) 335–370.

Honda, K., Vasconcelos, V. T. and Kubo, M. (1998) Language Primitives and Type Disciplines for Structured Communication-based Programming. In: Hankin, C. (ed.) ESOP'98. *Springer-Verlag Lecture Notes in Computer Science* **1381** 22–138.

Honda, K. and Yoshida, N. (1994) Replication in Concurrent Combinators. In: Hagiya, M. and Mitchell, J. C. (eds.) TACS'94. *Springer-Verlag Lecture Notes in Computer Science* **789** 786–805.

Levi, F. and Sangiorgi, D. (2003) Controlling Interference in Ambients. *Transactions on Programming Languages and Systems* **25** (1) 1–69.

Lhoussaine, C. and Sassone, V. (2004) A Dependently Typed Ambient Calculus. In: Schmidt, D. (ed.) ESOP'04. *Springer-Verlag Lecture Notes in Computer Science* **2986** 171–187.

Merro, M. and Hennessy, M. (2006) Bisimulation Congruences in Safe Ambients. *ACM Transactions on Programming Languages and System.* **28** (2) 290–330.

Merro, M. and Sassone, V. (2002) Typing and Subtyping Mobility in Boxed Ambients. In: Brim, L., Jancar, P., Kretinsky, M. and Kucera, A. (eds.) CONCUR'02. *Springer-Verlag Lecture Notes in Computer Science* **2421** 304–320.

Milner, R. (1993) The Polyadic π-calculus: a Tutorial. In: Bauer, F. L., Brauer, W. and Schwichtenberg, H. (eds.) Logic and Algebra of Specification. *NATO ASI Series F: Computer and Systems Sciences*, Springer-Verlag **94** 203–246.

Milner, R. and Sangiorgi, D. (1992) Barbed Bisimulation. In: Kuich, W. (ed.) ICALP '92. *Springer-Verlag Lecture Notes in Computer Science* **623** 685–695.

Necula, G. C. (1997) Proof-Carrying Code. In: Jones, N. D. (ed.) *POPL'97*, ACM Press 106–119.

Sangiorgi, D. and Walker, D. (2002) *The π-calculus*, Cambridge University Press.

Teller, D., Zimmer, P. and Hirschkoff, D. (2002) Using Ambients to Control Resources. In: Brim, L., Jančar, P., Křetínský, M. and Kučera, A. (eds.) CONCUR'02. *Springer-Verlag Lecture Notes in Computer Science* **2421** 288–303.

Zimmer, P. (2000) Subtyping and Typing Algorithms for Mobile Ambients. In: Tiuryn, J. (ed.) FOSSACS'00. *Springer-Verlag Lecture Notes in Computer Science* **1784** 375–390.