Boxed Ambients with Communication Interfaces [⋆]

Eduardo Bonelli[1], Adriana Compagnoni[1],
Mariangiola Dezani-Ciancaglini[2], and Pablo Garralda[1]

[1] Stevens Institute of Technology, U.S.A.
[2] Università di Torino, Italy

Abstract. We define **BACI** *(Boxed Ambients with Communication Interfaces)*, an ambient calculus allowing a liberal communication policy. Each ambient carries its local view of the topic of conversation (the type of the information being exchanged) with parents and children that will condition where it is allowed to stay or migrate to and which ambients may be allowed to enter it. The topic of conversation view of ambients can dynamically change during migration. **BACI** is flexible enough to allow different topics of conversation between an ambient and different parents, without compromising type-safety: it uses port names for communication and ambient names for mobility. Capabilities and co-capabilities exchange port names and run-time typing information to control mobility. We show the type-soundness of **BACI** proving that it satisfies the subject reduction property. Moreover we study its behavioural semantics by means of a labelled transition system.

1 Introduction

In an ambient calculus one can distinguish two forms of dynamic behavior: *communication* and *migration* [10]. By communication we mean the exchange of information between processes possibly 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 may enable or disable communication and vice-versa.

In calculi such as BA and NBA, and those in [6,23,7,11], 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 among the processes within an ambient. In typed ambient calculi, communication is controlled by types, and the type of information being exchanged is often called *topic of conversation (TOC)*. For example, if an ambient sends the number 3 to its parent, we can say that the TOC is Int.

Furthermore, notice that migration (entering or exiting an ambient) changes the parent of an ambient. The existing typed mobile ambient calculi fix a TOC for communication with the parent for each ambient, and even if migrating changes the parent the TOC remains fixed.

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 fixed-TOC-with-parent restriction allowing an ambient to change TOC when changing parents and enabling straightforward design of ambients that need to exchange information of different types with different ambients.

[⋆] This research was partially supported by the EU within the FET - Global Computing initiative, project DART ST-2001-33477, by MURST Cofin'02 project McTati, 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.

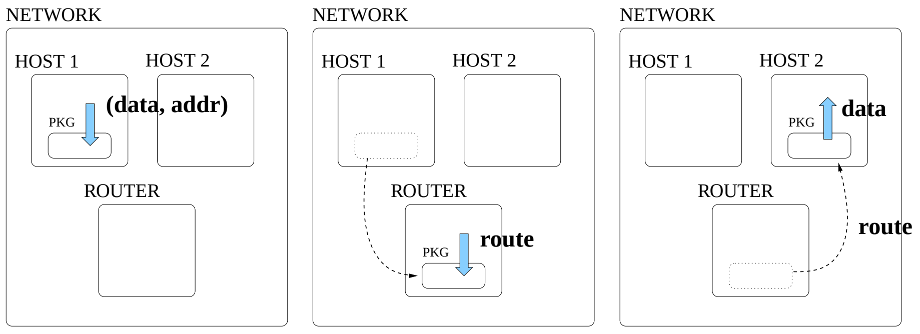

Fig. 1. Example of an ambient using different TOCs with different parents

Consider the example in Figure 1, where HOST 1 needs to send **data** to HOST 2, but HOST 1 does not know where HOST 2 is located. However, HOST 1 knows the location (**addr**) of a ROUTER that can forward to HOST 2 the packet (PKG) containing the **data**. Assuming this, HOST 1 spawns the packet and forwards the data to be transported along with the location of the router. Next, the packet moves inside the ROUTER where it obtains the **route** to HOST 2. Finally, using that **route** the packet reaches HOST 2 and delivers the **data**. Notice that the PKG ambient uses three different communication types with its three different parents (i.e. 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. **BACI**'s new features offer more flexibility to the designer to deliver more natural specifications.

Ports and Names The communication with a child ambient is often labeled with the ambient's name (named communication):

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

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.

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

In order to allow different TOCs with different parents, **BACI** introduces named communication with parents.

As we mentioned earlier, we can distinguish between two forms of dynamic behavior: communication and mobility. Previous calculi used ambient names to specify both communication and mobility. However, adding the possibility to communicate with different parents in different types stressed the difference between them. Our calculus, **BACI**, induces the separation of concerns by using names only for mobility and introducing the concept of *ports* for communication. Therefore to migrate to an ambient we use its name, but to exchange information with it we use its port. In this framework, an ambient n with communication port c is written $n[c\| \cdots]$, so for example:

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

The introduction of communication ports naturally leads to associate TOCs to ports, instead of associating TOCs to ambient names as usual. There is no global knowledge on this association: each ambient has its *local view* which can dynamically increase with relocation. An ambient n with port c and local view Γ is written $n[\Gamma \| c \| \cdots]$. Then our last example becomes:

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

To sum up, in **BACI** each ambient comes equipped with *its own local communication interface*. A communication interface consists of

– a *communication port* used by the other ambients to communicate with the current one and
– a *local view* associating topics of conversation to parent and child ports.

Communication interfaces are required for communication across ambient boundaries. In order for communication across ambient boundaries to take place a pair of communication ports must be coupled: a *sending communication port* (provided by the sending ambient) and a *receiving communication port* (provided by the receiving ambient). The following $(\text{INPUT} \downarrow\text{-}\uparrow)$ reduction rule models one form of communication across ambient boundaries (the other one consists in reversing the roles of sending ambient and receiving ambient)

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

The sending ambient is n and the receiving ambient is m. The communication interface of m consists of the communication port c_m and the local view Γ_m. Likewise, the communication interface of n consists of the communication port c_n and the local view Γ_n. In order for communication to succeed, the type of the information sent through the sending communication port (c_n) must coincide with the one expected by the receiving communication port (c_m). More precisely, the message $\tilde{M}$ in n should have some type $\tilde{\varphi}$ that coincides with the one that the ambient m is expecting.

Non-determinism The same port name may be used by different ambients. For example, in

$$n[\Gamma_n \| c \| \langle q \rangle^{\uparrow c_p} \mid P] \mid m[\Gamma_m \| c \| \langle \overline{\mathsf{in}} \rangle^{\uparrow c_p} \mid Q]$$

both ambients use the same port name c as part of their communication interfaces. Moreover, both ambients intend to communicate with a parent port c_p, but in different ways: ambient n wants to send the name q while ambient m wants to send the capability $\overline{\mathsf{in}}$ (which is used to allow an ambient to enter). In other words, for n, c_p is a parent port for exchanging ambient names, and Γ_n contains $\mathsf{amb}^{\uparrow c_p}$, and for m, c_p is a parent port for exchanging capabilities and Γ_m contains $\mathsf{cap}^{\uparrow c_p}$. As a consequence, only one of these ambients would be able to enter a host ambient p with communication port c_p willing to communicate with them. For example, in

$$p[\Gamma_p \| c_p \| (x : \varphi)^{\downarrow c}.R \mid \overline{\mathsf{in}}.\mathbf{0}] \mid n[\Gamma_n \| c \| \langle q \rangle^{\uparrow c_p} \mid \mathsf{in}\ p.\mathbf{0}] \mid m[\Gamma_m \| c \| \langle \overline{\mathsf{in}} \rangle^{\uparrow c_p} \mid \mathsf{in}\ p.\mathbf{0}]$$

if $\varphi = \downarrow c(\Gamma_p) = \mathsf{amb}$, then n is allowed to enter and if $\varphi = \downarrow c(\Gamma_p) = \mathsf{cap}$ then m is allowed to enter. Both of the above examples are typable in our system assuming that the port of the host ambient is different from c_p.

Related Work Modeling the word wide web requires the space and the mobility in space as new dimensions of computing. In the first proposals, i.e. in the *Dπ-calculus* [18] and in the language *Klaim* [14], the structure of locations is flat. Instead the *Mobile Ambient (MA)* calculus [10] deals with a hierarchical structure of locations (called ambients). An interesting *core model* generalising many of the available calculi and languages has been developed inside the Mikado project [4].

Many variants of MA have been designed: for a tutorial see [15]. A crucial choice in all these calculi is the form of *interaction* between processes in different ambients. In the original calculus [10] interaction is only local to an ambient, and in order for processes in different ambients to communicate, at least one of the ambients' boundaries have to be dissolved. In [10,20,5,22,1], the open capability dissolves the ambient boundary. The calculus **M3** [13,12], allows general process mobility. In *Boxed Ambients (BA)* [6,23,7], parents and children can communicate as in the *Seal* calculus [11]. Our calculus, **BACI**, follows this last protocol.
The *co-actions* (first introduced in [20], and then used with modifications in [5,22,23,7]) require also the agreement of the "passive" ambients involved in mobility. The co-actions of **BACI** in which port names are communicated are inspired by those of [7]: there the communication involves only the name of the entering ambient.

Ambient calculi are often typed: the types assure behavioural properties concerning communication, mobility, resource access, security, etc. [8,20,9,5,1,6,22,23,7,21]. To our knowledge before **BACI** only the calculi of [5] and [12] 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 [16] we define an *operational semantics with types*, which is simpler that a fully-fledged *typed operational semantics* in the sense that we only need to check agreement between the local views upon mobility. In some sense it can be seen as special case of *proof carrying code* [24]. The local type information in **BACI** can dynamically increase with ambient movements: something similar happens in the version of Dπ-calculus considered in [19].

Behavioural types [1,2] look mainly as computational traces: they allow polymorphic communications. **BACI**'s communication interfaces also are a permissive tool for typing non local communications. In [2] the type of communication with the parent changes when communication takes place. However, they do not have named communication with the parent, and cannot express the fact that communication with different parents has different types as in our last example.

Paper Plan The rest of this paper is structured as follows. Section 2 introduces the syntax of the calculus and its operational semantics. Section 3 presents the type system. Section 4 studies a reduction barbed congruence and a labeled transition system (LTS). The bisimilarity induced by the LTS is shown to be sound with respect to the congruence. Some congruence laws are also identified. Section 5 discusses two extended examples highlighting the features of **BACI**. Finally, we conclude and suggest further research.

2 The Calculus

2.1 Terms and Types

The syntax for types and terms in **BACI** is given in Table 1. Notice that **BACI** is a typed calculus and as such by process we always mean a well-formed process according to the rules of Table 6.We assume two disjoint denumerable sets of variables one for name, capability and message variables, and the other for port variables. We use $m, n, o, p, q \ldots$ for ambient name constants and $x, y, z, \ldots$ for ambient name variables, while α, β range over both ambient constants and ambient variables. Communication port constants are written $c, c_n, \ldots$ and $v, v', \ldots$ are used for communication port variables, while γ is either a communication port constant or variable. The expressions $\mathsf{inC}(v : \rho)\ \alpha$, $\mathsf{outC}(v : \rho)\ \alpha$, $\overline{\mathsf{inC}}(v : \rho), \overline{\mathsf{outC}}(v : \rho)$ are *binders* for v in prefixes and processes.

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

Table 1. Syntax of **BACI**

The process **0** is the null process; $P_1 \mid P_2$ denotes the parallel composition of processes P_1 and P_2; $(\boldsymbol{\nu} n)P$ is the usual restriction operator that binds all free occurrences of n in P; ! is the replication operator. The expression $\pi.P$ denotes the process that performs an action (or a co-action) π and then continues with P. The π actions 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 η. The location η is the location of the communication exchange: $\uparrow \gamma$ denotes communication with a parent having the port γ, $\downarrow \gamma$ denotes communication with a child with port γ, and $\star$ denotes local communication.

The information exchanged in input/output are tuples of messages. Each message can be either an ambient name, or a (co)-capability[1]. The ambient names received as messages, can substitute a variable ambient name in an ambient constructor or in a capability. Capabilities and co-capabilities constitute the mobility actions and co-actions: the capabilities allow an ambient to enter another ambient α, using $\mathsf{in}\ \alpha$; or to exit an ambient α, using $\mathsf{out}\ \alpha$. In order to be executed, each capability must be matched with its respective co-capability: in with $\overline{\mathsf{in}}$ and out with $\overline{\mathsf{out}}$. Both capabilities and co-capabilities can be sent as messages. A single (co)-capability or several (co)-capabilities forming a path may be sent.

Beside 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

[1] The types of messages can be easily extended to handle basic types such as integer or boolean without any technical problems, but that is omitted here for the sake of simplicity.

$!P \equiv P \mid !P$	(Struct Rep Par)
$(\boldsymbol{\nu} n)(\boldsymbol{\nu} m)P \equiv (\boldsymbol{\nu} m)(\boldsymbol{\nu} n)P$	(Struct Res Res)
$(\boldsymbol{\nu} n)(P \mid Q) \equiv P \mid (\boldsymbol{\nu} n)Q,\ \text{if } n \notin \mathrm{fn}(P)$	(Struct Res Par)
$(\boldsymbol{\nu} n)m[\Gamma_m \Vert c_m \Vert P] \equiv m[\Gamma_m \Vert c_m \Vert (\boldsymbol{\nu} n)P],\ \text{if } n \neq m$	(Struct Res Amb)
$\mathsf{in}\ n.P \equiv \mathsf{inC}(v : \mathsf{shh})\ n\ .P,\ \text{if } v \notin \mathrm{fv}(P)$	(Struct In InC)
$\mathsf{out}\ n.P \equiv \mathsf{outC}(v : \mathsf{shh})\ n\ .P,\ \text{if } v \notin \mathrm{fv}(P)$	(Struct Out OutC)
$\overline{\mathsf{in}}.P \equiv \overline{\mathsf{inC}}(v : \mathsf{shh}).P,\ \text{if } v \notin \mathrm{fv}(P)$	(Struct Co-in Co-inC)
$\overline{\mathsf{out}}.P \equiv \overline{\mathsf{outC}}(v : \mathsf{shh}).P,\ \text{if } v \notin \mathrm{fv}(P)$	(Struct Co-Out Co-OutC)
$(C.D).P \equiv C.D.P$	(Struct .)

Table 2. Structural Congruence

port variable that is bound at execution time with the port of the counterpart ambient involved in the mobility action. Because **BACI** uses port names exclusively for communication and ambient names only for mobility, knowing the name of an ambient is not enough to establish a communication, the port associated with that ambient must be known as well.

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 communication type ρ. If the communication type is shh, there is no exchange of information, otherwise an exchange tuple of basic types is used to indicate exchange of information of that type. The types used on these actions and co-actions must be compatible, their relation will be established by the operational semantics rules.

An ambient is written $\alpha[\Gamma \| c \| P]$ where α is an ambient name constant or an ambient name variable and P the enclosed process. The local view Γ is a finite set of located types, i.e. of exchange tuple types decorated with a location. The local view is used to specify the communication type of the enclosed process P. The local view together with the communication port c constitute the communication interface of the ambient.

Terms differing only in the names of their bound variables are considered equal. Furthermore, *Barendregt's convention* [3] is assumed: all variables are pairwise distinct and distinct from all free variables. This avoids cluttering the presentation with conditions on the names of variables in order to prevent variable clash and variable capture.

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 $(\ \mid\ , \mathbf{0})$ is a commutative monoid and the axioms of Table 2 are satisfied. The definition is standard except for the rules in the second group (where $\mathrm{fv}(P)$ is the set of free variables occurring in P). They state that the (co-)capabilities $\mathsf{in}\ n, \mathsf{out}\ n, \overline{\mathsf{in}}, \overline{\mathsf{out}}$ may be identified with particular instances of the prefixes "port enter/exit" and "allow port enter/exit". The rationale here is that these prefixes behave as the corresponding (co-)capabilities when they cancel the communicated port name (condition $v \notin \mathrm{fv}(P)$) and no topic of conversation is communicated (i.e. the communication type is shh). We cannot, however, do away completely with the (co-)capabilities since, in contrast to the aforementioned port prefixes, they may be sent as messages.

Join of communication types

$$\rho \sqcup \rho' = \begin{cases} \rho & \text{if } \rho = \rho' \text{ or } \rho' = \mathsf{shh}, \\ \rho' & \text{if } \rho = \mathsf{shh}, \\ \bot & \text{otherwise.} \end{cases}$$

Preorder on communication types

$$\rho \preceq \rho' \text{ iff } \rho \sqcup \rho' = \rho'$$

Addition of located types to local views

$$\mathit{\Gamma} \oplus \rho^{\eta} = \begin{cases} \mathit{\Gamma} & \text{if } \rho = \mathsf{shh}, \\ \mathit{\Gamma}, \rho^{\eta} & \text{otherwise.} \end{cases}$$

Application of locations

– to located types

$$\eta(\tau) = \begin{cases} \rho & \text{if } \tau = \rho^{\eta}, \\ \mathsf{shh} & \text{otherwise.} \end{cases}$$

– to local views

$$\begin{aligned} \eta(\emptyset) &= \mathsf{shh} \\ \eta(\mathit{\Gamma}) &= \textstyle\bigsqcup_{\tau \in \mathit{\Gamma}} \eta(\tau) \end{aligned}$$

Location substitution

– for prefixes

$$\pi\{\eta := \eta'\} = \begin{cases} (x_1 : \varphi_1, \ldots, x_k : \varphi_k)^{\eta'} & \text{if } \pi = (x_1 : \varphi_1, \ldots, x_k : \varphi_k)^{\eta}, \\ \langle M_1, \ldots, M_k \rangle^{\eta'} & \text{if } \pi = \langle M_1, \ldots, M_k \rangle^{\eta}, \\ \pi & \text{otherwise.} \end{cases}$$

– for processes

- $\mathbf{0}\{\eta := \eta'\} = \mathbf{0}$
- $(P_1 \mid P_2)\{\eta := \eta'\} = P_1\{\eta := \eta'\} \mid P_2\{\eta := \eta'\}$
- $((\boldsymbol{\nu} n)P)\{\eta := \eta'\} = (\boldsymbol{\nu} n)P\{\eta := \eta'\}$
- $(!P)\{\eta := \eta'\} = !P\{\eta := \eta'\}$
- $(\alpha[\mathit{\Gamma}_\alpha \| c_\alpha \| P])\{\eta := \eta'\} = \alpha[\mathit{\Gamma}_\alpha \| c_\alpha \| P]$
- $(\pi.P)\{\eta := \eta'\} = \pi\{\eta := \eta'\}.P\{\eta := \eta'\}$

Table 3. Operations on Locations and Local Views

Mobility

(ENTER)

$$n[\mathit{\Gamma}_n \| c_n \| \mathsf{inC}(v:\rho)\ m\ .P_1 \mid P_2] \mid m[\mathit{\Gamma}_m \| c_m \| \overline{\mathsf{inC}}(v':\rho').Q_1 \mid Q_2]$$
$$\longrightarrow$$
$$m[\mathit{\Gamma}_m \oplus \rho'^{\downarrow c_n} \| c_m \| n[\mathit{\Gamma}_n \oplus \rho^{\uparrow c_m} \| c_n \| P_1\{\uparrow v := \uparrow c_m\} \mid P_2] \mid Q_1\{\downarrow v' := \downarrow c_n\} \mid Q_2]$$
$$\text{if } \uparrow c_m(\mathit{\Gamma}_n) \sqcup \rho \preceq \downarrow c_n(\mathit{\Gamma}_m) \sqcup \rho'$$

(EXIT)

$$p[\mathit{\Gamma}_p \| c_p \| n[\mathit{\Gamma}_n \| c_n \| m[\mathit{\Gamma}_m \| c_m \| \mathsf{outC}(v:\rho)\ n\ .P_1 \mid P_2] \mid Q] \mid \overline{\mathsf{outC}}(v':\rho').R_1 \mid R_2]$$
$$\longrightarrow$$
$$p[\mathit{\Gamma}_p \oplus \rho'^{\downarrow c_m} \| c_p \| m[\mathit{\Gamma}_m \oplus \rho^{\uparrow c_p} \| c_m \| P_1\{\uparrow v := \uparrow c_p\} \mid P_2] \mid n[\mathit{\Gamma}_n \| c_n \| Q] \mid R_1\{\downarrow v' := \downarrow c_m\} \mid R_2]$$
$$\text{if } \uparrow c_p(\mathit{\Gamma}_m) \sqcup \rho \preceq \downarrow c_m(\mathit{\Gamma}_p) \sqcup \rho'$$

Communication

(LOCAL)

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

(INPUT $\downarrow$-$\uparrow$)

$$m[\mathit{\Gamma}_m \| c_m \| (\tilde{x}:\tilde{\varphi})^{\downarrow c_n}.P \mid n[\mathit{\Gamma}_n \| c_n \| \langle \tilde{M} \rangle^{\uparrow c_m}.Q \mid R] \mid S]$$
$$\longrightarrow$$
$$m[\mathit{\Gamma}_m \| c_m \| P\{\tilde{x} := \tilde{M}\} \mid n[\mathit{\Gamma}_n \| c_n \| Q \mid R] \mid S]$$

(OUTPUT $\downarrow$-$\uparrow$)

$$m[\mathit{\Gamma}_m \| c_m \| \langle \tilde{M} \rangle^{\downarrow c_n}.P \mid n[\mathit{\Gamma}_n \| c_n \| (\tilde{x}:\tilde{\varphi})^{\uparrow c_m}.Q \mid R] \mid S]$$
$$\longrightarrow$$
$$m[\mathit{\Gamma}_m \| c_m \| P \mid n[\mathit{\Gamma} \| c_n \| Q\{\tilde{x} := \tilde{M}\} \mid R] \mid S]$$

Structural Rules

(STRUCT)

$$\frac{P \equiv P', \quad P' \longrightarrow Q', \quad Q' \equiv Q}{P \longrightarrow Q}$$

(CONTEXT)

$$\frac{P \longrightarrow Q}{\mathbf{E}\{P\} \longrightarrow \mathbf{E}\{Q\}}$$

Evaluation Contexts

$$\mathbf{E} \ ::= \ \{\cdot\} \ \mid \ \mathbf{E}|P \ \mid \ P|\mathbf{E} \ \mid \ (\boldsymbol{\nu} n)\mathbf{E} \ \mid \ \alpha[\mathit{\Gamma} \| c \| \mathbf{E}]$$

Table 4. Operational Semantics

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 some definitions given in Table 3.

The join of communication types ($\rho \sqcup \rho'$) and the preorder on them ($\rho \preceq \rho'$) simply state that shh is smaller than any other communication type.

The addition of an expression ρ^η to a local view Γ ($\Gamma \oplus \rho^\eta$) changes Γ only if ρ is a tuple of basic types, i.e. if $\rho \neq \mathsf{shh}$. Notice that only in this last case ρ^η is a located type.

The application of a location η to a located type τ ($\eta(\tau)$) returns either the communication type of τ or shh according to whether the location of τ is η or not. The application of a location η to a local view Γ ($\eta(\Gamma)$) is the join of the applications of η to the located types in Γ. So $\eta(\Gamma)$ is different from $\perp$ only if Γ contains at most one located type whose location is η; $\eta(\Gamma)$ is shh when Γ contains no located type whose location is η.

The location substitution ($\{\eta := \eta'\}$) replaces locations as superscripts of input and output prefixes. They propagate on processes in the standard way but they never cross ambient boundaries.

The *mobility rules* consist of (ENTER) and (EXIT). Since they may both be explained along similar lines, we discuss only the former. The (ENTER) rule allows an ambient n to enter a sibling ambient m. Once n has entered m, communication may take place. However, this requires that n directs its messages through m's communication port (namely c_m) and, likewise, that m directs its messages through n's communication port (namely c_n). Since n may not know the name of m's communication port, the capability $\mathsf{inC}(v : \rho)\ m$ and the co-capability $\overline{\mathsf{inC}}(v' : \rho')$ provide port variables v and v' for such communication ports to be made available to the interested parties.

Note that the type of the information that may be exchanged on the port is also provided at run-time so that n and m may use compatible topics of conversation. Indeed, as a consequence of n entering m, the local views of both ambients are updated. Since this takes place at run-time, appropriate checks are required in order to guarantee that such extensions are sound. This is the role of the condition $\uparrow c_m(\Gamma_n) \sqcup \rho \preceq \downarrow c_n(\Gamma_m) \sqcup \rho'$. This condition may be explained as follows:

1. First notice that $\uparrow c_m(\Gamma_n) \sqcup \rho$ must be defined, since $\preceq$ is a partial relation between communication types. This implies that the new communication type ρ with location $\uparrow c_m$ to be added to the local view Γ_n is compatible with any existing located type in Γ_n. A similar comment applies to $\downarrow c_n(\Gamma_m) \sqcup \rho'$.
2. Assuming $\rho_1 = \uparrow c_m(\Gamma_n) \sqcup \rho$ and $\rho_2 = \downarrow c_n(\Gamma_m) \sqcup \rho'$, the condition $\rho_1 \preceq \rho_2$ checks to see whether n and m agree on a topic of conversation. Note that n may safely avoid listening to m but not vice-versa.

The following example shows why children can safely avoid to listen to parents but not vice-versa, i.e. why the condition $\rho_2 \preceq \rho_1$ is unsafe.

Take the process

$$p[\emptyset \| c \| \overline{\mathsf{inC}}(v_1 : \mathsf{shh}) \mid \overline{\mathsf{inC}}(v_2 : \mathsf{amb}).\langle q \rangle^{\downarrow v_2}]$$
$$\mid m[\mathsf{cap}^{\uparrow c} \| c' \| \mathsf{inC}(v : \mathsf{cap})\ p\ \mid (x : \mathsf{cap})^{\uparrow c}.P] \mid n[\emptyset \| c' \| \mathsf{inC}(v' : \mathsf{amb})\ p\ ]$$

Using rule (ENTER) with the pre-order condition reversed this process might reduce first to

$$p[\emptyset \| c \| \overline{\mathsf{inC}}(v_2 : \mathsf{amb}).\langle q \rangle^{\downarrow v_2} \mid m[\mathsf{cap}^{\uparrow c} \| c' \| (x : \mathsf{cap})^{\uparrow c}.P]] \mid n[\emptyset \| c' \| \mathsf{inC}(v' : \mathsf{amb})\ p\ ]$$

and then to

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

and in this last process a wrong communication of an ambient name when a capability is expected could occur.

Notice that with the sound rule (ENTER) the initial process only reduces to

$$p[\mathsf{amb}^{\downarrow c'} \| c \| \overline{\mathsf{inC}}(v_1 : \mathsf{shh}) \mid \langle q \rangle^{\downarrow c'} \mid n[\mathsf{amb}^{\uparrow c} \| c' \|]] \mid m[\mathsf{cap}^{\uparrow c} \| c' \| \mathsf{inC}(v : \mathsf{cap})\, p \mid (x : \mathsf{cap})^{\uparrow c}.P]$$

i.e. only ambient n is allowed to go inside ambient p.

Thanks to the structural congruence between prefixes and (co-)capabilities, the standard enter and exit rules are mimicked by (ENTER) and (EXIT), respectively. For example, the following (STANDARD ENTER) rule:

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

may be simulated by (ENTER). Indeed, if v, v' are any port variables such that $v \notin \mathrm{fv}(P_1)$ and $v \notin \mathrm{fv}(Q_1)$, then since $\mathsf{in}\ m.P_1 \equiv \mathsf{inC}(v : \mathsf{shh})\ m\ .P_1$ and $\overline{\mathsf{in}}.Q_1 \equiv \overline{\mathsf{inC}}(v : \mathsf{shh}).Q_1$, by applying (ENTER) we obtain

$$m[\Gamma_m \oplus \mathsf{shh}^{\downarrow c_n} \| c_m \| n[\Gamma_n \oplus \mathsf{shh}^{\uparrow c_m} \| c_n \| P_1\{\uparrow v := \uparrow c_m\} \mid P_2] \mid Q_1\{\downarrow v' := \downarrow c_n\} \mid Q_2]$$

Note that $P_1\{\uparrow v := \uparrow c_m\} = P_1$ and $Q_1\{\downarrow v' := \downarrow c_n\} = Q_1$. Also, $\Gamma_m \oplus \mathsf{shh}^{\downarrow c_n} = \Gamma_m$ and $\Gamma_n \oplus \mathsf{shh}^{\uparrow c_m} = \Gamma_n$ (cf. Table 3).

As for the *communication rules*, the local communications are standard, while the parent-child communications require the knowledge of the partner communication port, as already discussed in the introduction.

3 Typing Rules

The typing environment is very simple: it says if a variable stands for an ambient name or a capability.

Environments

$\Sigma ::= \emptyset$		empty environment
$\mid$	$\Sigma, x : \varphi$	environment

In the sequel, we only consider typing environments that assign a unique type to each name in its domain. The typing rules define two judgements:

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

The typing rules for the first judgements appear in Table 5.

In contrast to other systems, the (amb) rule assigns an ambient name the constant type amb rather than a more informative type as in the majority of systems [15]. Indeed, more informative types presuppose the availability of (global) information on the type of ambients. In our setting based on local views the only assumption we make is that we can identify an ambient name when we see one.

The (cap-in -out) and (cap-$\overline{\mathsf{in}}$ -$\overline{\mathsf{out}}$) rules are also simpler than in formulations based on global knowledge of the communication types of ambients, since the corresponding control is delegated to run-time.

The (AXIOM) and (cap-COMP) rules are standard.
The rules defining the judgement $\Sigma \vdash_c P : \Gamma$ are given in Table 6.

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

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

Table 5. Well-formed Messages

Regarding the typing rule (**0**), since **0** does not interact with its host it may be typed under a communication interface consisting of any port name c and interface view Γ proviso Γ is *ok*, i.e. $\eta(\Gamma) \neq \bot$ for all η, and each port variable occurs at most once in Γ.

The rule for replication (PROC-REP) is standard, however (PROC-RES) is not. Usually, the name n together with its type is assumed to belong to the global environment Σ. However, in our local setting all we know is that n is an ambient name.

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

The typing rule (PROC-CAP) reveals that all what is known about a capability is that it is just a capability. Since we rely only on local information we shall relegate the correct use of capabilities at 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, if P is well-formed under the assumption that the host ambient has local view $\Gamma \oplus \rho^{\uparrow v}$. Thus, the prefix $\mathsf{inC}(v : \rho)\ \alpha$ allows its host ambient to extend its local knowledge and hence be ready to communicate with arbitrary ambients willing to enter. Note that this prefix binds the free occurrences of the port variable v in P. The typing of the other prefixes mentioned in rules (PROC-INC-OUTC) and (PROC-COINC-COOUTC) is similar. The difference between these two rules is that in the first one the process shall communicate with a new host ambient, whereas in the second one the process shall communicate with a newly entering child ambient.

The (PROC-INPUT) and (PROC-OUTPUT) request that the type of the information that is exchanged together with its location must belong to the local view of the host ambient.

The (PROC-AMB) rule may be interpreted as follows. In order for $\beta[\Gamma' \| c' \| P]$ 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 satisfied that:

1. process P is well-formed under a host ambient whose communication interface consists of port c' and local view Γ', where β must be an ambient name or an ambient variable;
2. either $\beta[\Gamma' \| c' \| P]$ does not communicate with its host ambient or the type of the information exchange between it and its host ambient must be the same (condition $\uparrow c(\Gamma') \preceq\ \downarrow c'(\Gamma)$);
3. no free port variables should occur in Γ', i.e. Γ' should be *closed*;
4. the local view Γ of the host ambient must be *ok*.

The example considered on page 9 shows why we do not allow $\beta[\Gamma' \| c' \| P]$ to offer a communication to its host ambient when the host ambient does not communicate along

(**0**)
$$\frac{\Gamma\ ok}{\Sigma \vdash_c \mathbf{0} : \Gamma}$$

(PROC-REP)
$$\frac{\Sigma \vdash_c P : \Gamma}{\Sigma \vdash_c !P : \Gamma}$$

(PROC-RES)
$$\frac{\Sigma \vdash_c P : \Gamma}{\Sigma \vdash_c (\boldsymbol{\nu} n)P : \Gamma}$$

(PROC-COMP)
$$\frac{\Sigma \vdash_c P_1 : \Gamma \quad \Sigma \vdash_c P_2 : \Gamma}{\Sigma \vdash_c P_1 \mid P_2 : \Gamma}$$

(PROC-CAP)
$$\frac{\Sigma \vdash C : \mathsf{cap} \quad \Sigma \vdash_c P : \Gamma}{\Sigma \vdash_c C.P : \Gamma}$$

(PROC-INC-OUTC)
$$\frac{\Sigma \vdash_c P : \Gamma \oplus \rho^{\uparrow v} \quad \pi \in \{\mathsf{inC}(v : \rho)\ \alpha\ , \mathsf{outC}(v : \rho)\ \alpha\ \}}{\Sigma \vdash_c \pi.P : \Gamma}$$

(PROC-COINC-COOUTC)
$$\frac{\Sigma \vdash_c P : \Gamma \oplus \rho^{\downarrow v} \quad \pi \in \{\overline{\mathsf{inC}}(v : \rho), \overline{\mathsf{outC}}(v : \rho)\}}{\Sigma \vdash_c \pi.P : \Gamma}$$

(PROC-INPUT)
$$\frac{\Sigma, x_1 : \varphi_1, \ldots, x_k : \varphi_k \vdash_c P : \Gamma \quad (\varphi_1 \times \ldots \times \varphi_k)^\eta \in \Gamma}{\Sigma \vdash_c (x_1 : \varphi_1, \ldots, x_k : \varphi_k)^\eta.P : \Gamma}$$

(PROC-OUTPUT)
$$\frac{\Sigma \vdash_c P : \Gamma \quad \Sigma \vdash M_i : \varphi_i\ (1 \le i \le k) \quad (\varphi_1 \times \ldots \times \varphi_k)^\eta \in \Gamma}{\Sigma \vdash_c \langle M_1, \ldots, M_k \rangle^\eta.P : \Gamma}$$

(PROC-AMB)
$$\frac{\Sigma \vdash_{c'} P : \Gamma' \quad \Sigma \vdash \beta : \mathsf{amb} \quad \uparrow c(\Gamma') \preceq \downarrow c'(\Gamma) \quad \Gamma' \text{ is closed} \quad \Gamma \text{ is } ok}{\Sigma \vdash_c \beta[\Gamma' \| c' \| P] : \Gamma}$$

Table 6. Well-formed Processes

the port c'. In fact, the process

$$p[\emptyset \| c \| \overline{\mathsf{inC}}(v : \mathsf{amb}).\langle q \rangle^{\downarrow v} \mid m[\mathsf{cap}^{\uparrow c} \| c' \| (x : \mathsf{cap})^{\uparrow c}.P]] \mid n[\emptyset \| c' \| \mathsf{inC}(v' : \mathsf{amb})\ p\,]$$

is typable by replacing the condition $\uparrow c'(\Gamma) \preceq \downarrow c(\Gamma')$ to $\uparrow c(\Gamma') \preceq \downarrow c'(\Gamma)$ in rule (PROC-AMB).

The type system guarantees that communication inside ambients and across ambient boundaries never leads to type mismatches. This is formalized as:

Theorem 1 (Subject Reduction). *If $\Sigma \vdash_c P : \Gamma$ and $P \longrightarrow Q$, then $\Sigma \vdash_c Q : \Gamma$.*

4 Behavioral Semantics

In order to study the behavioral semantics of **BACI** we define an intuitive notion of barbed congruence [25,17] based on the unlabelled reduction semantics given in Table 4. We then introduce a labelled transition semantics inspired by [20,22,7] and state 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 the vast body of proof techniques to work in order to reason about barbed congruence. Note that in this short presentation we omit the global environment Σ, the host port c and the local view Γ over which the relations on well-formed processes are indexed by.

Since **BACI** has co-capabilities and allows parent-child communications there are several reasonable choices of barbs, among which we have:

$$\begin{array}{lrcl}
(1) & P\downarrow^1_n & \triangleq & P \equiv (\boldsymbol{\nu}\tilde{m})(n[\Gamma\|c\|\overline{\mathsf{in}}.Q \mid R] \mid S) \\
(2) & P\downarrow^2_n & \triangleq & P \equiv (\boldsymbol{\nu}\tilde{m})(n[\Gamma\|c\|\overline{\mathsf{inC}}(v:\rho).Q \mid R] \mid S) \\
(3) & P\downarrow^3_{\langle c,c'\rangle} & \triangleq & P \equiv (\boldsymbol{\nu}\tilde{m})(n[\Gamma\|c\|(x_1:\varphi_1,\ldots,x_k:\varphi_k)^{\uparrow c'}.Q \mid R] \mid S) \\
(4) & P\downarrow^4_{\langle c,c'\rangle} & \triangleq & P \equiv (\boldsymbol{\nu}\tilde{m})(n[\Gamma\|c\|\langle M_1,\ldots,M_k\rangle^{\uparrow c'}.Q \mid R] \mid S)
\end{array}$$

provided that $n \notin \tilde{m}$ in (1) and (2). In order to *observe* whether a process P may interact with the environment via some ambient name n or via a pair of port names $\langle c, c'\rangle$, it must be placed in a context that presents an ambient that attempts to enter it or to communicate with it.

We write $P\Downarrow_n$ ($P\Downarrow_{\langle c,c'\rangle}$) if $P \Longrightarrow P'$ and $P'\downarrow_n$ ($P\downarrow_{\langle c,c'\rangle}$), where $\Longrightarrow$ is the reflexive and transitive closure of $\longrightarrow$.

The notions of observational congruence induced by the above definitions of barb are standard in ambient calculi.

Definition 1. *A relation $\mathcal{R}$ is reduction closed if $P\mathcal{R}Q$ and $P \longrightarrow P'$ imply the existence of some Q' such that $Q \Longrightarrow Q'$ and $P'\mathcal{R}Q'$. $\mathcal{R}$ is barb preserving if $P\mathcal{R}Q$ and $P\downarrow_n$ ($P\downarrow_{\langle c,c'\rangle}$) imply $Q\Downarrow_n$ ($Q\Downarrow_{\langle c,c'\rangle}$).*

Definition 2 (Reduction Barbed Congruence). *Reduction barbed congruence is the largest equivalence relation that is preserved by contexts and, when restricted to closed processes, is reduction closed and barb preserving. Let then $\cong_i$ be the reduction barbed congruence relation from choosing the notion of observation as in (i) above (with $i \in$ [1..4]).*

Notice that since we only consider processes which are well-formed, a relation $\mathcal{R}$ is preserved by contexts if $P\mathcal{R}Q$ and $C[P]$ well-formed imply $C[Q]$ well-formed and $C[P]\mathcal{R}C[Q]$, for all processes P, Q and contexts $C[\cdot]$.

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

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

Proof. We need to show that all barbs imply each other. This can be accomplished, as usual, by exhibiting a corresponding context. For instance, to see that $\cong_2$ implies $\cong_3$ use the context $C[\cdot] = m[\Gamma\|c'\|[\cdot] \mid \langle\tilde{M}\rangle^{\downarrow c}.\overline{\mathsf{inC}}(v:\rho)]$, and note that for all P such that m is fresh in P one has $P\Downarrow^3_{\langle c,c'\rangle}$ if and only if $C[P]\Downarrow^2_m$. A suitable context to show that $\cong_4$ implies $\cong_1$ is $C[\cdot] = p[\Gamma\|c\|(\boldsymbol{\nu}q)(q[\emptyset\|c_q\|[\cdot] \mid m[\emptyset\|c_m\|\mathsf{in}\ n.\mathsf{out}\ n.\mathsf{out}\ q] \mid \overline{\mathsf{out}}]) \mid \overline{\mathsf{out}}.\langle\tilde{M}\rangle^{\uparrow c'}]$, and similarly for the other cases.

Notice that processes with different types can be distinguished irrespective of their purely behavioural properties. This means that if two processes P and Q cannot be typed with the same $\mathit{\Gamma}$ (w.r.t. a given c), they cannot be congruent. In fact, if $\mathit{\Sigma} \vdash_c P : \mathit{\Gamma}$ but $\mathit{\Sigma} \vdash_c Q : \mathit{\Gamma}$ does not hold we can find a context $C[\cdot]$ such that $C[P]$ is a (well-formed) process while $C[Q]$ is not. A suitable context is simply $n[\mathit{\Gamma}'\|c\|\pi.[\cdot]]$, where $\mathit{\Gamma}'$ is the maximum subset of $\mathit{\Gamma}$ which is closed, and π contains (in any order) exactly the set of actions $\{\mathsf{inC}(v : \rho)\ m \mid \rho^{\uparrow v}\}$ and the set of co-actions $\{\overline{\mathsf{inC}}(v : \rho) \mid \rho^{\downarrow v}\}$. For that reason, type equality is not required *per se* for the definition of barbed congruence.

4.1 Algebraic Laws

This section presents some algebraic laws that better portray the semantics of processes in **BACI**. These and other laws can be proved by means of the labelled bisimilarity developed in the next section.

The laws holding in **BACI** which deal with mobility are very similar to those true for the NBA calculus [7], so we will not discuss them.

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

$$n[\mathit{\Gamma}_n\|c_n\|m[\mathit{\Gamma}_m\|c_m\|(\tilde{x} : \tilde{\varphi})^{\uparrow c}.P] \mid Q] \cong n[\mathit{\Gamma}_n\|c_n\|Q]$$

$$n[\mathit{\Gamma}_n\|c_n\|m[\mathit{\Gamma}_m\|c_m\|\langle\tilde{M}\rangle^{\uparrow c}.P] \mid Q] \cong n[\mathit{\Gamma}_n\|c_n\|Q]$$

In NBA a communication parent-child can be forced only if it is the only active process inside both ambients. In **BACI** instead there can be other active processes provided that they do not know the port name of the communication partner and some ambient names do not occur in some processes and/or they are restricted. The conditions on port names avoid interfering communications and the conditions on ambient names avoid interfering movements. In particular in the third group of equivalencies R cannot contain m since otherwise an ambient inside R could exit m and communicate the port c_m to process S. More precisely we have:
if c_m does not occur in R

$$\begin{gathered}(\boldsymbol{\nu} n)(n[\mathit{\Gamma}_n\|c_n\|m[\mathit{\Gamma}_m\|c_m\|\langle\tilde{M}\rangle^{\uparrow c_n}.P] \mid (\tilde{x} : \tilde{\varphi})^{\downarrow c_m}.Q \mid R])\\ \cong\\ (\boldsymbol{\nu} n)(n[\mathit{\Gamma}_n\|c_n\|m[\mathit{\Gamma}_m\|c_m\|P] \mid Q\{\tilde{x} := \tilde{M}\} \mid R])\end{gathered}$$

$$\begin{gathered}(\boldsymbol{\nu} n)(n[\mathit{\Gamma}_n\|c_n\|m[\mathit{\Gamma}_m\|c_m\|(\tilde{x} : \tilde{\varphi})^{\uparrow c_n}.P] \mid \langle\tilde{M}\rangle^{\downarrow c_m}.Q \mid R])\\ \cong\\ (\boldsymbol{\nu} n)(n[\mathit{\Gamma}_n\|c_n\|m[\mathit{\Gamma}_m\|c_m\|P\{\tilde{x} := \tilde{M}\}] \mid Q \mid R])\end{gathered}$$

if c_n and n do not occur in R

$$\begin{gathered}n[\mathit{\Gamma}_n\|c_n\|(\boldsymbol{\nu} m)(m[\mathit{\Gamma}_m\|c_m\|\langle\tilde{M}\rangle^{\uparrow c_n}.P \mid R]) \mid (\tilde{x} : \tilde{\varphi})^{\downarrow c_m}.Q]\\ \cong\\ (\boldsymbol{\nu} m)(n[\mathit{\Gamma}_n\|c_n\|m[\mathit{\Gamma}_m\|c_m\|P \mid R]) \mid Q\{\tilde{x} := \tilde{M}\}]\end{gathered}$$

$$\begin{gathered}n[\mathit{\Gamma}_n\|c_n\|(\boldsymbol{\nu} m)(m[\mathit{\Gamma}_m\|c_m\|(\tilde{x} : \tilde{\varphi})^{\uparrow c_n}.P \mid R]) \mid \langle\tilde{M}\rangle^{\downarrow c_m}.Q]\\ \cong\\ n[\mathit{\Gamma}_n\|c_n\|(\boldsymbol{\nu} m)(m[\mathit{\Gamma}_m\|c_m\|P\{\tilde{x} := \tilde{M}\} \mid R]) \mid Q]\end{gathered}$$

Pre-prefixes	κ	$::=$	$\mathsf{in}\ n \mid \mathsf{out}\ n \mid \overline{\mathsf{in}} \mid \mathsf{inC}(c:\rho)\ n \mid \mathsf{outC}(c:\rho)\ n \mid \overline{\mathsf{inC}}(c:\rho)$
Prefixes	μ	$::=$	$\kappa \mid \langle\tilde{M}\rangle^{\eta} \mid \overline{\mathsf{outC}}(c:\rho) \mid \overline{\mathsf{out}}$
Labels	ξ	$::=$	$\tau \mid \mu \mid \langle\tilde{-}\rangle^{\eta} \mid n[\mathit{\Gamma}_n \Vert c_n \Vert \kappa] \mid \tilde{M}\ \mathsf{get}\ (c_m, c_n) \mid \mathsf{put}\ (c_m, c_n)$
Concretions	K	$::=$	$(\boldsymbol{\nu}\tilde{p})\langle\!\langle P\rangle\!\rangle Q \mid (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M}\rangle\!\rangle P$
Outcomes	O	$::=$	$P \mid K$

Table 7. Labels and Outcomes

if c_m, m do not occur in S and c_n, n, m do not occur in R

$$(\boldsymbol{\nu}n)(n[\mathit{\Gamma}_n\Vert c_n\Vert(\boldsymbol{\nu}m)(m[\mathit{\Gamma}_m\Vert c_m\Vert\langle\tilde{M}\rangle^{\uparrow c_n}.P \mid R]) \mid (\tilde{x}:\tilde{\varphi})^{\downarrow c_m}.Q \mid S])$$
$$\cong$$
$$(\boldsymbol{\nu}n)(\boldsymbol{\nu}m)(n[\mathit{\Gamma}_n\Vert c_n\Vert m[\mathit{\Gamma}_m\Vert c_m\Vert P \mid R]) \mid Q\{\tilde{x}:=\tilde{M}\} \mid S])$$

$$(\boldsymbol{\nu}n)(n[\mathit{\Gamma}_n\Vert c_n\Vert(\boldsymbol{\nu}m)(m[\mathit{\Gamma}_m\Vert c_m\Vert(\tilde{x}:\tilde{\varphi})^{\uparrow c_n}.P \mid R]) \mid \langle\tilde{M}\rangle^{\downarrow c_m}.Q \mid S])$$
$$\cong$$
$$(\boldsymbol{\nu}n)(n[\mathit{\Gamma}_n\Vert c_n\Vert(\boldsymbol{\nu}m)(m[\mathit{\Gamma}_m\Vert c_m\Vert P\{\tilde{x}:=\tilde{M}\} \mid R]) \mid Q \mid S])$$

4.2 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 characterization of reduction barbed congruence in terms of labelled bisimulation. The LTS is given in Tables 8, 9 and 10. These tables define the labelled transition relation

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

where P is a process, ξ is a label and O is an "outcome". Labels and outcomes are defined in Table 7.

An outcome may be a process or a *concretion*. Concretions are required for dealing with transitions of components of the system that interact with the environment in order 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 the part of the system that interacts with the environment. For example, to complete an $\mathsf{in}\ n$ transition, the sibling ambient which hosts the entering one must be requested from the context. Likewise, 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 cases, Q represents the remaining part of the process that is not affected by the transition.

The structural congruence relation for concretions is obtained by extending the homonymous relation for processes with the following axioms and rules:

$$\begin{array}{lcl}
(\boldsymbol{\nu}r)((\boldsymbol{\nu}\tilde{p})\langle\!\langle P\rangle\!\rangle Q) & \equiv & (\boldsymbol{\nu}r,\tilde{p})\langle\!\langle P\rangle\!\rangle Q \\
(\boldsymbol{\nu}r)((\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M}\rangle\!\rangle P) & \equiv & (\boldsymbol{\nu}r,\tilde{p})\langle\!\langle \tilde{M}\rangle\!\rangle P
\end{array}$$

$$\begin{array}{lcl}
P \equiv P' \text{ and } Q \equiv Q' & \Longrightarrow & (\boldsymbol{\nu}\tilde{p})\langle\!\langle P\rangle\!\rangle Q \equiv (\boldsymbol{\nu}\tilde{p})\langle\!\langle P'\rangle\!\rangle Q' \\
P \equiv P' & \Longrightarrow & (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M}\rangle\!\rangle P \equiv (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M}\rangle\!\rangle P'
\end{array}$$

Also, we use the following notational conventions:

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

(PATH)
$$\frac{M_1.(M_2.P) \xrightarrow{\xi} P'}{(M_1.M_2).P \xrightarrow{\xi} P'}$$

(CAPC)
$$\frac{\zeta \in \{\mathsf{inC}, \mathsf{outC}\}}{\zeta(v : \rho)\ n\ .P \xrightarrow{\zeta(c:\rho)\ n} P\{\uparrow v := \uparrow c\}}$$

(CO-CAPC)
$$\frac{\zeta \in \{\overline{\mathsf{inC}}, \overline{\mathsf{outC}}\}}{\zeta(v : \rho).P \xrightarrow{\zeta(c:\rho)} P\{\downarrow v := \downarrow c\}}$$

(IN-OUT)
$$\frac{P \xrightarrow{\xi} P' \quad \xi \in \{\mathsf{in}\ n, \mathsf{out}\ n\}}{m[\Gamma_m \| c_m \| P] \xrightarrow{m[\Gamma_m \| c_m \| \xi]} \langle\!\langle m[\Gamma_m \| c_m \| P'] \rangle\!\rangle \mathbf{0}}$$

(CO-IN)
$$\frac{P \xrightarrow{\xi} P' \quad \xi \in \{\overline{\mathsf{in}}, \overline{\mathsf{inC}}(c : \rho)\}}{m[\Gamma_m \| c_m \| P] \xrightarrow{m[\Gamma_m \| c_m \| \xi]} \langle\!\langle P' \rangle\!\rangle \mathbf{0}}$$

(INC-OUTC)
$$\frac{P \xrightarrow{\xi} P' \quad \xi \in \{\mathsf{inC}(c : \rho)\ n, \mathsf{outC}(c : \rho)\ n\} \quad \Gamma_m \oplus \rho^{\uparrow c}\ \mathit{ok}}{m[\Gamma_m \| c_m \| P] \xrightarrow{m[\Gamma_m \| c_m \| \xi]} \langle\!\langle m[\Gamma_m \oplus \rho^{\uparrow c} \| c_m \| P'] \rangle\!\rangle \mathbf{0}}$$

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

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

(GET)
$$\frac{P \xrightarrow{(\tilde{M})^{\uparrow c}} P_1}{m[\Gamma_m \| c_m \| P] \xrightarrow{\tilde{M}\ \mathsf{get}\ (c_m, c)} m[\Gamma_m \| c_m \| P_1]}$$

(PUT)
$$\frac{P \xrightarrow{\langle - \rangle^{\uparrow c}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P_1}{m[\Gamma_m \| c_m \| P] \xrightarrow{\mathsf{put}\ (c_m, c)} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle m[\Gamma_m \| c_m \| P_1]}$$

Table 8. Commitments: Visible transitions

– $((\boldsymbol{\nu}\tilde{p})\langle\!\langle P \rangle\!\rangle Q) \mid R = (\boldsymbol{\nu}\tilde{p})\langle\!\langle P \rangle\!\rangle (Q \mid R)$
– $((\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P) \mid R = (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle (P \mid R)$

The transitions are inspired by those of NBA [7]. The τ transitions for message exchanges are (τ-EXCHANGE) for local exchange and (τ-PUT) and (τ-GET) for non-local exchange. For example, in (τ-PUT) the directed output action towards the child ambient must be met by a corresponding input action from the child. Rule (GET) makes sure that this input action is executed inside some ambient whose local communication port coincides with the one specified in the output action.

The τ transitions for mobility are (τ-ENTER), (τ-ENTERC), (τ-EXIT), (τ-EXITC). Since these are similar in spirit we shall discuss only (τ-ENTERC). Rule (τ-ENTERC) is in charge of synchronizing two actions, namely the request of an ambient to enter a host ambient with the action witnessing the approval (by means of an appropriate co-capability) on the part of the host ambient. Therefore, the label of the first action is $n[\Gamma_n \| c_n \| \mathsf{inC}(c_m : \rho)\ m\ ]$ while that of the second is $m[\Gamma_m \| c_m \| \overline{\mathsf{inC}}(c_n : \rho')]$.

The former tells of the name and local interface information of the moving ambient and the latter does the same for the host ambient. The process that actually moves is represented by P_1 in the concretion resulting from the first action while Q_1 represents the process that shall run alongside the visiting ambient. The processes P_2 and Q_2 are the sub-components of P and Q, respectively, that do not participate in the movement. Note that the third premise of the rule (τ-ENTERC) corresponds to the dynamic type checking that we discussed for reduction.

As expected, unlabelled reduction and labelled reduction coincide. Both items of Theorem 2 are proved by induction on the derivation of the antecedent. Moreover, item 2 requires the following lemma that relates labelled reduction and structural congruence, for the case when the derivation is obtained using the rule (STRUCT).

Lemma 2. *If $P \xrightarrow{\xi} O$ and $P \equiv Q$, then there exists O' such that $Q \xrightarrow{\xi} O'$ and $O \equiv O'$.*

Theorem 2.

1. *If $P \xrightarrow{\tau} P'$, then $P \longrightarrow P'$.*
2. *If $P \longrightarrow P'$, then $P \xrightarrow{\tau}\equiv P'$, where $\xrightarrow{\tau}\equiv$ denotes the composition of the relations $\xrightarrow{\tau}$ and $\equiv$.*

By comparing the notion of observability (cf. the definition of barbs) with the rules of Table 8 and in particular with rule (CO-IN) one can easily see that a name is observable iff at least one of the two actions $n[\Gamma\|c\|\overline{\mathsf{in}}]$ or $n[\Gamma\|c\|\overline{\mathsf{inC}}(c' : \rho)]$ can be performed. In particular,

Lemma 3. *$P\downarrow_n^1$ iff $P \xrightarrow{n[\Gamma\|c\|\overline{\mathsf{in}}]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle Q\rangle\!\rangle R$ for some Γ, c, $\tilde{p}$, Q, R.*

A similar observation applies to rules (GET), (PUT) and the observability of pairs of port names (cf. barbs (3) and (4) above). Thanks to Lemma 1 we only need to consider one notion of barb.

4.3 Full Bismilarity 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* [22] for those labelled transitions of Table 8 that produce a concretion as an outcome.

The HO-transitions are given in Table 11. In these transitions we use richer labels obtained by adding to the previous labels ξ a new component which can be of one of the following five shapes:

– P;
– $[\Gamma\|c\|P]$;
– $n[\Gamma\|c\|P]$;
– $[\Gamma\|c\|P]n[\Gamma'\|c'\|Q]$;
– $n[\Gamma\|c\|P] \mid Q$.

This component describes the minimum contribution of the context necessary to fire the transition. For example in rule (HO OUT) the context must provide both the 3 components (local view, port and process) of the ambient n from which the process P_1 exits and in which the process P_2 remains and the whole ambient q in which the process P_1 enters.

For HO transitions we get the following version of Lemma 3:

(τ-ENTER)

$$\frac{P \xrightarrow{n[\mathit{\Gamma}_n \| c_n \| \mathsf{in}\ m]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1\rangle\!\rangle P_2 \quad Q \xrightarrow{m[\mathit{\Gamma}_m \| c_m \| \overline{\mathsf{in}}]} (\boldsymbol{\nu}\tilde{q})\langle\!\langle Q_1\rangle\!\rangle Q_2 \quad \uparrow c_m(\mathit{\Gamma}_n) \preceq \downarrow c_n(\mathit{\Gamma}_m)}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p},\tilde{q})(m[\mathit{\Gamma}_m \| c_m \| Q_1 \mid P_1] \mid P_2 \mid Q_2)}$$

(τ-ENTERC)

$$\frac{\begin{array}{c} P \xrightarrow{n[\mathit{\Gamma}_n \| c_n \| \mathsf{inC}(c_m : \rho)\ m\ ]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1\rangle\!\rangle P_2 \quad Q \xrightarrow{m[\mathit{\Gamma}_m \| c_m \| \overline{\mathsf{inC}}(c_n : \rho')]} (\boldsymbol{\nu}\tilde{q})\langle\!\langle Q_1\rangle\!\rangle Q_2 \\ \uparrow c_m(\mathit{\Gamma}_n) \sqcup \rho \preceq \downarrow c_n(\mathit{\Gamma}_m) \sqcup \rho' \end{array}}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p},\tilde{q})(m[\mathit{\Gamma}_m \oplus \rho'^{\downarrow c_n} \| c_m \| Q_1 \mid P_1] \mid P_2 \mid Q_2)}$$

(τ-EXIT)

$$\frac{P \xrightarrow{n[\mathit{\Gamma}_n \| c_n \| \mathsf{out}\ m]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1\rangle\!\rangle P_2 \quad Q \xrightarrow{\overline{\mathsf{out}}} Q_1 \quad \uparrow c_q(\mathit{\Gamma}_n) \preceq \downarrow c_n(\mathit{\Gamma}_q)}{q[\mathit{\Gamma}_q \| c_q \| m[\mathit{\Gamma}_m \| c_m \| P] \mid Q] \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p})(q[\mathit{\Gamma}_q \| c_q \| m[\mathit{\Gamma}_m \| c_m \| P_2] \mid P_1 \mid Q_1])}$$

(τ-EXITC)

$$\frac{P \xrightarrow{n[\mathit{\Gamma}_n \| c_n \| \mathsf{outC}(c_q : \rho)\ m\ ]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1\rangle\!\rangle P_2 \quad Q \xrightarrow{\overline{\mathsf{outC}}(c_n : \rho')} Q_1 \quad \uparrow c_q(\mathit{\Gamma}_n) \sqcup \rho \preceq \downarrow c_n(\mathit{\Gamma}_q) \sqcup \rho'}{q[\mathit{\Gamma}_q \| c_q \| m[\mathit{\Gamma}_m \| c_m \| P] \mid Q] \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p})(q[\mathit{\Gamma}_q \oplus \rho'^{\downarrow c_n} \| c_q \| m[\mathit{\Gamma}_m \| c_m \| P_2] \mid P_1 \mid Q_1])}$$

(τ-EXCHANGE)

$$\frac{P \xrightarrow{(\tilde{M})^\star} P_1 \quad Q \xrightarrow{\langle - \rangle^\star} (\boldsymbol{\nu}\tilde{q})\langle\!\langle \tilde{M}\rangle\!\rangle Q_1}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{q})(P_1 \mid Q_1)}$$

(τ-PUT)

$$\frac{P \xrightarrow{\langle - \rangle^{\downarrow c_m}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M}\rangle\!\rangle P_1 \quad Q \xrightarrow{\tilde{M}\ \mathsf{get}\ (c_n, c_m)} Q_1}{n[\mathit{\Gamma}_n \| c_n \| P \mid Q] \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p}) n[\mathit{\Gamma}_n \| c_n \| P_1 \mid Q_1]}$$

(τ-GET)

$$\frac{P \xrightarrow{(\tilde{M})^{\downarrow c_m}} P_1 \quad Q \xrightarrow{\mathsf{put}\ (c_m, c_n)} (\boldsymbol{\nu}\tilde{q})\langle\!\langle \tilde{M}\rangle\!\rangle Q_1}{n[\mathit{\Gamma}_n \| c_n \| P \mid Q] \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{q})(n[\mathit{\Gamma}_n \| c_n \| P_1 \mid Q_1])}$$

Table 9. Commitments: τ transitions

(PAR)
$$\frac{P \xrightarrow{\xi} O}{P \mid Q \xrightarrow{\xi} O \mid Q}$$

(RES)
$$\frac{P \xrightarrow{\xi} O \quad n \notin \mathrm{fn}(\xi)}{(\boldsymbol{\nu} n)P \xrightarrow{\xi} (\boldsymbol{\nu} n)O}$$

(τ-AMB)
$$\frac{P \xrightarrow{\tau} P'}{n[\Gamma \| c \| P] \xrightarrow{\tau} n[\Gamma \| c \| P']}$$

(REPL)
$$\frac{\pi.P \xrightarrow{\xi} O}{!\pi.P \xrightarrow{\xi} !\pi.P \mid O}$$

Table 10. Commitments: Structural transitions

Lemma 4. *$P \downarrow_n^1$ iff $P \xrightarrow{n[\Gamma \| c \| \overline{\mathsf{in}}] m[\emptyset \| c' \| \mathbf{0}]} (\boldsymbol{\nu}\tilde{p})(n[\Gamma \| c \| Q \mid m[\emptyset \| c' \| \mathbf{0}]] \mid R)$ for all m, c' and for some Γ, c, $\tilde{p}$, Q, R.*

As last step towards defining labelled bisimilarity, let Λ denote the set of labels that includes both the first order labels defined in Tables 8 and 10 and the HO ones of Table 11. In the following notational convention we let λ range over Λ. Let $\Longrightarrow$ denote the reflexive and transitive closure of $\xrightarrow{\tau}$.

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

Definition 3 (Bisimulation). *A symmetric relation $\mathcal{R}$ over closed processes is a bisimulation if $P \mathcal{R} Q$ and $P \xrightarrow{\lambda} P'$ imply there exists Q' such that*

- *$Q \overset{\hat{\lambda}}{\Longrightarrow} Q'$ and*
- *$P' \mathcal{R} Q'$.*

Two closed processes P and Q are bisimilar, written $P \approx_c Q$, if $P \mathcal{R} Q$ for some bisimulation $\mathcal{R}$.

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

Definition 4 (Full Bisimilarity). *Two processes P and Q are fully bisimilar, written $P \approx Q$, if $P\mathsf{s} \approx_c Q\mathsf{s}$ for every closing substitution s that respects types.*

Following the proof scheme of [22,7] we can show that full bisimilarity is preserved by context.

Theorem 3. *Full bisimilarity is a congruence.*

Moreover from Lemma 4 it follows that:

Lemma 5. *Full bisimilarity is barb preserving over closed processes.*

Proof. Suppose P, Q are closed processes, $P \approx_c Q$ and $P \downarrow_n^1$.
By Lemma 4 $P \xrightarrow{n[\Gamma \| c \| \overline{\mathsf{in}}] m[\emptyset \| c' \| \mathbf{0}]} P'$ for all m, c' and some Γ, c, P'. As a consequence $Q \overset{n[\Gamma \| c \| \overline{\mathsf{in}}] m[\emptyset \| c' \| \mathbf{0}]}{\Longrightarrow} Q'$ for some Q'. In particular, there is a Q'' such that $Q \Longrightarrow Q'' \xrightarrow{n[\Gamma \| c \| \overline{\mathsf{in}}] m[\emptyset \| c' \| \mathbf{0}]} Q'$. From Lemma 4 we deduce $Q'' \downarrow_n^1$ and hence $Q \Downarrow_n^1$, as required.

(HO IN)

$$\frac{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{in}\ n]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \uparrow c_n(\Gamma_m) \preceq \downarrow c_m(\Gamma_n)}{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{in}\ n][\Gamma_n \| c_n \| Q]} (\boldsymbol{\nu}\tilde{p})(n[\Gamma_n \| c_n \| P_1 \mid Q] \mid P_2)}$$

(HO INC)

$$\frac{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{inC}(c_n : \rho)\ n\ ]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \uparrow c_n(\Gamma_m) \sqcup \rho \preceq \downarrow c_m(\Gamma_n)}{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{inC}(c_n : \rho)\ n\ ][\Gamma_n \| c_n \| Q]} (\boldsymbol{\nu}\tilde{p})(n[\Gamma_n \| c_n \| P_1 \mid Q] \mid P_2)}$$

(HO CO-IN)

$$\frac{P \xrightarrow{n[\Gamma_n \| c_n \| \overline{\mathsf{in}}]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \uparrow c_n(\Gamma_m) \preceq \downarrow c_m(\Gamma_n)}{P \xrightarrow{n[\Gamma_n \| c_n \| \overline{\mathsf{in}}] m[\Gamma_m \| c_m \| Q]} (\boldsymbol{\nu}\tilde{p})(n[\Gamma_n \| c_n \| P_1 \mid m[\Gamma_m \| c_m \| Q]] \mid P_2)}$$

(HO CO-INC)

$$\frac{P \xrightarrow{n[\Gamma_n \| c_n \| \overline{\mathsf{inC}}(c_m : \rho)]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \uparrow c_n(\Gamma_m) \preceq \downarrow c_m(\Gamma_n) \sqcup \rho}{P \xrightarrow{n[\Gamma_n \| c_n \| \overline{\mathsf{inC}}(c_m : \rho)] m[\Gamma_m \| c_m \| Q]} (\boldsymbol{\nu}\tilde{p})(n[\Gamma_n \| c_n \| P_1 \mid m[\Gamma_m \| c_m \| Q]] \mid P_2)}$$

(HO OUT)

$$\frac{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{out}\ n]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \uparrow c_q(\Gamma_m) \preceq \downarrow c_m(\Gamma_q)}{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{out}\ n][\Gamma_n \| c_n \| Q] q[\Gamma_q \| c_q \| R]} (\boldsymbol{\nu}\tilde{p})(q[\Gamma_q \| c_q \| P_1 \mid n[\Gamma_n \| c_n \| P_2 \mid Q] \mid R])}$$

(HO OUTC)

$$\frac{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{outC}(c_n : \rho)\ n\ ]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \uparrow c_n(\Gamma_m) \sqcup \rho \preceq \downarrow c_m(\Gamma_n)}{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{outC}(c_n : \rho)\ n\ ][\Gamma_n \| c_n \| Q] q[\Gamma_q \| c_q \| R]} (\boldsymbol{\nu}\tilde{p})(q[\Gamma_q \| c_q \| P_1 \mid n[\Gamma_n \| c_n \| P_2 \mid Q] \mid R])}$$

(HO OUTPUT)

$$\frac{P \xrightarrow{\langle - \rangle^{\eta}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P' \quad \eta \in \{\star, \downarrow c\}}{P \xrightarrow{\langle - \rangle^{\eta} Q} (\boldsymbol{\nu}\tilde{p})(P' \mid Q\{\tilde{x} := \tilde{M}\})}$$

(HO PUT)

$$\frac{P \xrightarrow{\mathsf{put}\ (c_m, c_n)} (\boldsymbol{\nu}\tilde{q})\langle\!\langle \tilde{M} \rangle\!\rangle P'}{P \xrightarrow{\mathsf{put}\ (c_m, c_n) Q} (\boldsymbol{\nu}\tilde{q})(P' \mid Q\{\tilde{x} := \tilde{M}\})}$$

(HO OUTPUT$^{\uparrow}$)

$$\frac{P \xrightarrow{\langle - \rangle^{\uparrow c}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P'}{P \xrightarrow{\langle - \rangle^{\uparrow c} m[\Gamma_m \| c_m \| R] \mid Q} (\boldsymbol{\nu}\tilde{p})(m[\Gamma_m \| c_m \| P' \mid R] \mid Q\{\tilde{x} := \tilde{M}\})}$$

Table 11. Commitments: Higher-Order transitions

Finally, we prove the desired result that $\approx$ is contained in $\cong$.

Theorem 4 (Soundness of Full Bisimilarity). *If $P \approx Q$ then $P \cong Q$.*

Proof. It suffices to show that $\approx$ is a barbed bisimulation up to $\equiv$ (since then it follows that $\equiv\approx\equiv$ - ie. the composition of the relations $\equiv$, $\approx$ and $\equiv$ - is also a barbed bisimulation and $P \approx Q$ and $\equiv\approx\equiv\subseteq\cong$ imply $P \cong Q$). This follows from the fact that $\approx$:

1. is a congruence: Theorem 3.
2. is reduction closed on closed processes: Suppose P, Q are closed processes, $P \approx Q$ and $P \longrightarrow P'$. By Theorem 2, $P \xrightarrow{\tau}\equiv P'$. Since $P \approx Q$, there exists Q' such that $Q \Longrightarrow Q'$ and $P' \equiv\approx\equiv Q'$.
3. is barb preserving on closed processes: Lemma 5.

We conjecture incompleteness of $\approx$ for the same reason the authors of [7] conjecture incompleteness of the full bisimilarity arising from a similar LTS for NBA, namely the difficulty of finding a context which discriminates the label $\langle\tilde{M}\rangle^{\uparrow c}$. We conjecture also that a LTS for **BACI** inducing a complete full bisimilarity could be developed in the style of [7].

5 Examples

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

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

This allows sibling and nested ambients of α to freely enter and exit. Note that $\alpha^{\rightleftharpoons}$ allows to enter either ambients which do not communicate with α or ambients whose communication port name is already known by $\alpha^{\rightleftharpoons}$.

We convene not to write the types of the input variables since they are always clear from the context.

5.1 Remote printer

For this example we consider two networks (represented as ambients) called $n1$ and $n2$. Ambient $n1$ is the network where a client is located and $n2$ the one where a printer is located. Although the client ignores the path to the printer network, in $n1$ there is also a *router*, called $r1to2$, that knows the path to $n2$. For simplicity, we place $n1$ and $n2$ at the same nesting level inside a larger ambient, called $inter$. However, in general, $n1$ and $n2$ can be far from each other within the nesting hierarchies.

$\mathit{INTERNET} \triangleq inter^{\rightleftharpoons}[\emptyset\|c\|\mathit{N1} \mid \mathit{N2}]$

$\mathit{N1} \triangleq n1^{\rightleftharpoons}[\emptyset\|c_1\|\mathit{CLIENT} \mid \mathit{ROUTER}]$

$\mathit{N2} \triangleq n2^{\rightleftharpoons}[\emptyset\|c_2\|\mathit{PRINTER}]$

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 enter *ROUTER*. There, it shall receive the path to $n2$, where the printer is located. After reaching $n2$, the job enters the printer and communicate the data to be printed.

$$\begin{array}{l} \mathit{JOB}_{cl} \triangleq job[\mathit{\Gamma}_{job} \| c_j \| (d,p)^{\uparrow c_{cl}}.\mathsf{out}\ cl.\mathsf{in}\ r1to2.(route)^{\uparrow c_r}.route.\mathsf{in}\ p.\langle d \rangle^{\uparrow c_{pr}}] \\ \quad \text{where } \mathit{\Gamma}_{job} \triangleq \{(\mathsf{data} \times \mathsf{amb})^{\uparrow c_{cl}}, \mathsf{cap}^{\uparrow c_r}, \mathsf{data}^{\uparrow c_{pr}}\} \end{array}$$

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

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

$$\begin{array}{l} \mathit{CLIENT} \triangleq client1^{\rightleftharpoons}[\mathit{\Gamma}_{client1} \| c_{cl} \| \langle (d1, printer1) \rangle^{\downarrow c_j} \mid !\ \mathit{JOB}_{client1}] \\ \quad \text{where } \mathit{\Gamma}_{client1} \triangleq \{(\mathsf{data} \times \mathsf{amb})^{\downarrow c_j}\} \\ \mathit{ROUTER} \triangleq r1to2^{\rightleftharpoons}[\mathsf{cap}^{\downarrow c_j} \| c_r \| ! \langle (\mathsf{out}\ r1to2.\mathsf{out}\ n1.\mathsf{in}\ n2) \rangle^{\downarrow c_j}] \\ \mathit{PRINTER} \triangleq printer1^{\rightleftharpoons}[\mathsf{data}^{\downarrow c_j} \| c_{pr} \| !\ (d)^{\downarrow c_j}] \end{array}$$

After delivering its data the job ambient becomes inactive and useless. Using the algebraic properties we can show that $job[\mathit{\Gamma}_{job} \| c_j \| \mathbf{0}] \cong \mathbf{0}$ and therefore

$$\begin{array}{c} printer1^{\rightleftharpoons}[\mathsf{data}^{\downarrow c_j} \| c_{pr} \| !\ (d)^{\downarrow c_j} \mid job[\mathit{\Gamma}_{job} \| c_j \| \mathbf{0}]] \cong \\ printer1^{\rightleftharpoons}[\mathsf{data}^{\downarrow c_j} \| c_{pr} \| !\ (d)^{\downarrow c_j}] \end{array}$$

All these *garbage* ambients that accumulate inside the printer ambient can be safely discarded.

Since different ambients can have the same port name, more than one client can have port name c_{cl} and more than one server the port name c_{pr}, even if the ambients have different names. Moreover, we can add more clients and printers without changing *JOB* or *ROUTER*.

$$\begin{array}{l} \mathit{N1'} \triangleq n1^{\rightleftharpoons}[\emptyset \| c_1 \| \mathit{CLIENT} \mid \mathit{CLIENT'} \mid \mathit{ROUTER}] \\ \mathit{N2'} \triangleq n2^{\rightleftharpoons}[\emptyset \| c_2 \| \mathit{PRINTER} \mid \mathit{PRINTER'}] \\ \mathit{CLIENT'} \triangleq client2^{\rightleftharpoons}[\mathit{\Gamma}_{client2} \| c_{cl} \| \langle (d2, printer2) \rangle^{\downarrow c_j} \mid !\ \mathit{JOB}_{client2}] \\ \quad \text{where } \mathit{\Gamma}_{client2} \triangleq \{(\mathsf{data} \times \mathsf{amb})^{\downarrow c_j}\} \\ \mathit{PRINTER'} \triangleq printer2^{\rightleftharpoons}[\mathsf{data}^{\downarrow c_j} \| c_{pr} \| !\ (d)^{\downarrow c_j}] \end{array}$$

Having c_{cl} as a port name for all clients and c_{pr} for all servers allows any client to use any available printer, and not just a particular one as in the previous example. *CLIENT'* can also use *PRINTER'* by sending the message $\langle (d2, printer2) \rangle^{\downarrow c_j}$ to the spawned job.

The routing in the previous example was relatively simple with only one destination, only one route and only one router. How can we route a job to two different networks, for instance, $n2$ and $n3$? Here, we can get the destination network parameter from the client and use it to find the corresponding route. However, we need some mechanism to determine if we choose the route for $n2$ or the route for $n3$ depending on that parameter. There are no control flow primitives in the calculus similar to the test for equality found in π-calculus, for instance. Nevertheless, we can instruct the client to send the name of the router serving a given printer network (assuming we have a different router for each destination network), but the client would need to know the relation between the destination and the router that serves that destination. That is not very tidy. Besides, we would need to change the job interface, which seems to be very "natural" as it is. Another option is to take advantage of the locality of names and use the same name for both the destination network and the router serving the route to that destination. We now re-define the components of the system according to these new requirements:

$INTERNET \triangleq inter^{\rightleftharpoons}[\emptyset \| c_i \| N1" \mid N2" \mid N3]$

$N1" \triangleq n1^{\rightleftharpoons}[\emptyset \| c_1 \| CLIENT \mid CLIENT' \mid ROUTER \mid ROUTER']$

$N2" \triangleq n2^{\rightleftharpoons}[\emptyset \| c_2 \| PRINTER]$

$N3 \triangleq n3^{\rightleftharpoons}[\emptyset \| c_3 \| PRINTER']$

We moved one of the printers to network $n3$ to make the example more interesting. With this setting, any client located on $n1$ should be able to send a job to either the printer on $n2$ or the printer on $n3$. For this purpose, we change the name of the existing router (routing jobs to $n2$) and also add a new router that serves jobs heading to $n3$. Both routers have the same name as the routes they serve. Using the same names, we don't need to require the client to know the name of the routers. This gives us a clean and natural representation. We have to change the definition of *JOB* and the *CLIENTs* since in the previous example the router name was "hard-coded" and now is a parameter given by the client.

$JOB_{cl} \triangleq job[\Gamma_{job} \| c_j \| (d,p,n)^{\uparrow c_{cl}}.\mathsf{out}\ cl.\mathsf{in}\ n.(route)^{\uparrow c_r}.route.\mathsf{in}\ p.\langle data\rangle^{\uparrow c_{pr}}]$
  where $\Gamma_{job} \triangleq \{(\mathsf{data} \times \mathsf{amb} \times \mathsf{amb})^{\uparrow c_{cl}}, \mathsf{cap}^{\uparrow c_r}, \mathsf{data}^{\uparrow c_{pr}}\}$

$CLIENT \triangleq client1^{\rightleftharpoons}[\Gamma_{client1} \| c_{cl} \| \langle (d1, printer1, n2)\rangle^{\downarrow c_j} \mid !\ JOB_{client1}]$
  where $\Gamma_{client1} \triangleq \{(\mathsf{data} \times \mathsf{amb} \times \mathsf{amb})^{\downarrow c_j}\}$

$CLIENT' \triangleq client2^{\rightleftharpoons}[\Gamma_{client2} \| c_{cl} \| \langle (d2, printer2, n3)\rangle^{\downarrow c_j} \mid !\ JOB_{client2}]$
  where $\Gamma_{client2} \triangleq \{(\mathsf{data} \times \mathsf{amb} \times \mathsf{amb})^{\downarrow c_j}\}$

Finally, we change the name of the routers and we add the new router which has the same structure as the old one but with different route, of course.

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

$ROUTER' \triangleq n3^{\rightleftharpoons}[\mathsf{cap}^{\downarrow c_j} \| c_r \| !\langle(\mathsf{out}\ n3.\mathsf{out}\ n1.\mathsf{in}\ n3)\rangle^{\downarrow c_j}]$

The two orthogonal concepts of interfaces and names allow us to separate the input/output from the mobility concerns. We can use the interfaces to group several ambients with similar input/output abilities and, at the same time, we can keep each ambient identity by using different ambient names for each of them.

5.2 File servers cluster

This example represents some free download sites in which the user has a list of servers to choose for his download. However, for this example, we require that every time a customer requests a file download, the cluster designates one server from all the available servers in the cluster (i.e. 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 shutdown or power 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.

$CLUSTER \triangleq cluster^{\rightleftharpoons}[\Gamma_{clu} \| c_{clu} \| LOAD\_BAL \mid SRV1 \mid SRV2]$
  where $\Gamma_{clu} \triangleq \{\mathsf{amb} \times \mathsf{Filename}^{\downarrow c_{srv}}\}$

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

The *cluster* includes all the servers (only two in this example) 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. Notice that the client's communication port is not known in advance to the cluster and vice-versa. They are learned 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 spawning a messenger ambient called job. This job reaches the client and deliver the requested file. The acute reader will notice 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 "shutdown" a server.

$$SRVi \triangleq srvi^{\rightleftharpoons}[\mathit{\Gamma}_{srvi} \| c_{srv} \| !(on)^{\downarrow c_{pwr}}.\mathit{SERVE} \mid \mathit{PWR}]$$
$$\text{where } \mathit{\Gamma}_{srvi} \triangleq \{\mathsf{onMsg}^{\downarrow c_{pwr}}, \mathsf{amb} \times \mathsf{Filename}^{\uparrow c_{clu}}\}$$

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

$$\mathit{JOB} \triangleq job[\emptyset \| c_j \| \mathsf{out}\ srvi.\mathsf{inC}(v : \mathsf{data})\ clname\ .(\mathtt{file}(fname))^{\uparrow v}]$$

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

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 (and respond) to a request. Therefore, in order to shut a server down, the administrator should send a $POWER\_OFF$ message to that server.

$$\mathit{POWER\_OFF}(s) \triangleq \mathit{pwroff}^{\rightleftharpoons}[\emptyset \| c_{poff} \| \mathsf{in}\ cluster.\mathsf{in}\ s.\overline{\mathsf{in}}]$$

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

$$\mathit{pwroff}^{\rightleftharpoons}[\emptyset \| c_{\mathit{poff}} \| pwr[\mathsf{onMsg}^{\uparrow c_{srv}} \| c_{pwr} \| !\langle on \rangle^{\uparrow c_{srv}}]] \cong \mathbf{0}$$

and get rid of these *garbage* ambients.
Likewise, the administrator can restore the pwr ambient inside the server to "power on" that server.

$$\mathit{POWER\_ON}(s) \triangleq pwron^{\rightleftharpoons}[\emptyset \| c_{pon} \| \mathsf{in}\ cluster.\mathsf{in}\ s.\mathit{TURN\_ON}]$$

$$\mathit{TURN\_ON} \triangleq pwr[\mathsf{onMsg}^{\uparrow c_{srv}} \| c_{pwr} \| \mathsf{out}\ pwron \mid !\langle on \rangle^{\uparrow c_{srv}} \mid \mathsf{in}\ \mathit{poweroff}]$$

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, all of them are learned on execution. The only requirement is that the client is well behaved and it 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.

$$\mathit{CLIENT} \triangleq client^{\rightleftharpoons}[\mathit{\Gamma}_{cl} \| c_{client} \| \mathsf{inC}(v_{clu} : (\mathsf{amb} \times \mathsf{Filename}))\ cluster\ .$$
$$\langle client, \mathit{afilename} \rangle.\overline{\mathsf{inC}}(c_j : \mathsf{data}).(\mathsf{file})^{\downarrow \mathsf{c_j}}.\mathsf{P} \mid \mathsf{Q}]$$

This is the basic structure of a client ambient. The port name can be changed without restrictions. $\mathit{\Gamma}_{cl}$, *P* and *Q* can be anything that does not have conflicting types with the $cluster$ and job ambients . The whole configuration looks like this:

$$\mathit{SYSTEM} \triangleq \mathit{ADMIN} \mid \mathit{CLUSTER} \mid \mathit{CLIENTS}$$

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

6 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 the information that may be exchanged over parent and children ports. Besides 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 which it enters. Appropriate run-time checks are required so that the entering and the host ambient 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*. While in previous calculi communication with a parent was decided by the location of an ambient, in **BACI**, the communication with a parent is indexed by the parent's port, in a similar way in which communication with a child is usually indexed. This new named communication allows an ambient to communicate with different parents in different types (TOCs).
– *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, that was not possible.
– *Local typing*. Having different TOCs with different parents allows control over 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 so for mobility. Mobility is currently unrestricted and this poses the question if one might also include, in the local knowledge of an ambient, some indication of whether the ambient is allowed to move or not. Other items that warrant further work include: considering a restriction operator on port names, considering multiple ports (possibly taking dynamic port creation into account), matching and mismatching constructs and group types in order to impose access control.

References

1. Torben Amtoft, Assaf J. Kfoury, and Santiago M. Pericas-Geertsen. What are Polymorphically-Typed Ambients? In David Sands, editor, *ESOP'01*, volume 2028 of *LNCS*, pages 206–220, Berlin, 2001. Springer-Verlag.
2. Torben Amtoft, Henning Makholm, and Joe B. Wells. PolyA: True Type Polymorphism for Mobile Ambients. In *TCS'04*, 2004. to appear.
3. H.P. Barendregt. *The Lambda Calculus: its Syntax and Semantics*. Studies in Logic and the Foundations of Mathematics 103. North-Holland, Amsterdam, revised edition, 1984.
4. Gérard Boudol. 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, 2003.

5. Michele Bugliesi and Giuseppe Castagna. Behavioral Typing for Safe Ambients. *Computer Languages*, 28(1):61 – 99, 2002.
6. Michele Bugliesi, Giuseppe Castagna, and Silvia Crafa. Access Control for Mobile Agents: The Calculus of Boxed Ambients. *ACM Transactions on Programming Languages and Systems*, 26(1):57–124, 2004.
7. Michele Bugliesi, Silvia Crafa, Massimo Merro, and Vladimiro Sassone. Communication and Mobility Control in Boxed Ambients. To appear in *Information and Computation*. Extended and revised version of M. Bugliesi, S. Crafa, M. Merro, and V. Sassone. Communication Interference in Mobile Boxed Ambients. In FSTTCS'02, volume 2556 of LNCS, pages 71-84. Springer-Verlag, 2002.
8. Luca Cardelli, Giorgio Ghelli, and Andrew D. Gordon. Mobility Types for Mobile Ambients. In Jiri Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP'99*, volume 1644 of *LNCS*, pages 230–239, Berlin, 1999. Springer-Verlag.
9. Luca Cardelli, Giorgio Ghelli, and Andrew D. Gordon. Ambient Groups and Mobility Types. In Jan van Leeuwen, Osamu Watanabe, Masami Hagiya, Peter D. Mosses, and Takayasu Ito, editors, *TCS'00*, volume 1872 of *LNCS*, pages 333–347, Berlin, 2000. Springer-Verlag. Extended version to appear in Information and Computation, special issue on TCS'00.
10. Luca Cardelli and Andrew D. Gordon. Mobile Ambients. *Theoretical Computer Science*, 240(1):177–213, 2000. Special Issue on Coordination, Daniel Le Métayer Editor.
11. Giuseppe Castagna and Jan Vitek. Seal: A Framework for Secure Mobile Computations. In Henri E. Bal, Boumediene Belkhouche, and Luca Cardelli, editors, *Internet Programming Languages*, volume 1686 of *LNCS*, pages 47–77, Berlin, 1999. Springer-Verlag.
12. Mario Coppo, Mariangiola Dezani-Ciancaglini, Elio Giovannetti, and Rosario Pugliese. Dynamic and Local Typing for Mobile Ambients. In *TCS'04*, 2004. to appear.
13. Mario Coppo, Mariangiola Dezani-Ciancaglini, Elio Giovannetti, and Ivano Salvo. M3: Mobility Types for Mobile Processes in Mobile Ambients. In James Harland, editor, *CATS'03*, volume 78 of *ENTCS*. Elsevier, 2003.
14. Rocco De Nicola, GianLuigi Ferrari, and Rosario Pugliese. Klaim: a Kernel Language for Agents Interaction and Mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, 1998.
15. Elio Giovannetti. Ambient Calculi with Types: a Tutorial. In Corrado Priami, editor, *Global Computing - Programming Environments, Languages, Security and Analysis of Systems*, volume 2874 of *LNCS*, Berlin, 2003. Springer-Verlag.
16. Healfdene Goguen. Typed operational semantics. In Mariangiola Dezani-Ciancaglini and Gordon Plotkin, editors, *TLCA'95*, volume 902 of *LNCS*, pages 186–200, Berlin, 1995. Springer-Verlag.
17. Andrew D. Gordon and Luca Cardelli. Equational properties of mobile ambients. *Mathematical Structures in Computer Science*, 13(3):371–408, 2003.
18. Mattew Hennessy and James Riely. Resource Access Control in Systems of Mobile Agents. *Information and Computation*, 173:82–120, 2002.
19. Matthew Hennessy, Massimo Merro, and Julian Rathke. Towards a behavioural theory of access and mobility control in distributed system (extended abstract). In Andrew D. Gordon, editor, *FOSSACS'03*, volume 2620 of *LNCS*, pages 282–299, Berlin, 2003. Springer-Verlag.
20. Francesca Levi and Davide Sangiorgi. Controlling Interference in Ambients. *Transactions on Programming Languages and Systems*, 25(1):1–69, 2003.
21. Cédric Lhoussaine and Vladimiro Sassone. A Dependently Typed Ambient Calculus. In David Schmidt, editor, *ESOP'04*, volume 2986 of *LNCS*, pages 171–187, Berlin, 2004. Springer-Verlag.
22. Massimo Merro and Matthew Hennessy. Bisimulation Congruences in Safe Ambients. In Neil D. Jones and Xavier Leroy, editors, *POPL'02*, pages 71–80, New York, 2002. ACM Press.
23. Massimo Merro and Vladimiro Sassone. Typing and Subtyping Mobility in Boxed Ambients. In Lubos Brim, Petr Jancar, Mojmir Kretinsky, and Antonin Kucera, editors, *CONCUR'02*, volume 2421 of *LNCS*, pages 304–320, Berlin, 2002. Springer-Verlag.
24. George C. Necula. Proof-carrying code. In Neil D. Jones, editor, *POPL'97*, pages 106–119. ACM Press, 1997.
25. Davide Sangiorgi and Robin Milner. The problem of "Weak Bisimulation up to". In Walter R. Cleaveland, editor, *CONCUR'92*, volume 630 of *LNCS*, pages 32–46, Berlin, 1992. Springer-Verlag.