Bisimulation Congruences in Safe Ambients

Massimo Merro and Matthew Hennessy

Abstract. We study a variant of Levi and Sangiorgi's Safe Ambients (SA) enriched with *passwords* (SAP). In SAP by managing passwords, for example generating new ones and distributing them selectively, an ambient may now program who may migrate into its computation space, and when. Moreover in SAP an ambient may provide different services depending on the passwords exhibited by its incoming clients.

We give an *lts* based operational semantics for SAP and a labelled *bisimulation* based equivalence which is proved to coincide with barbed congruence.

We use our notion of bisimulation to prove a set of algebraic laws which are subsequently exploited to prove more significant examples.

1 Introduction

The calculus of *Mobile Ambients*, abbreviated MA, has been introduced in [6] as a novel process calculus for describing *mobile agents*. The term

$$n[P]$$

represents an agent, or *ambient*, named n, executing the code P. Intuitively $n[P]$ represents a bounded and protected space in which the computation P can take place. In turn P may contain other ambients, may effect communications, or may exercise *capabilities*, which allow entry to or exit from named ambients. Thus ambient names, such as n, are used to control access to the ambient's computation space and may be dynamically created as in the Picalculus, [12], using the construct $\boldsymbol{\nu} nP$; here knowledge of n is restricted to P. For example the system

$$k[\,\mathtt{in}\langle n\rangle.R_1 \mid R_2\,] \mid n[\,\mathtt{open}\langle k\rangle.P \mid m[\,\mathtt{out}\langle n\rangle.Q_1 \mid Q_2\,]\,]$$

contains two ambients, k and n, running concurrently. The first, k, has, at least, the capability to migrate into n, by virtue of its capability $\mathtt{in}\langle n\rangle$. The second, n, contains a sub-ambient $m[\ldots]$, in addition to the the capability $\mathtt{open}\langle k\rangle$, which allows the opening of any ambient named k which migrates into the computation space of n. If k exercises its capability to enter n then the system will have the structure

$$n[\; k[\ldots] \mid \mathtt{open}\langle k\rangle.P \mid m[\ldots]\;]$$

COGS, University of Sussex, UK. Research funded by EPSRC grant GR/M71169.

in which, now, n may exercise its capability to dissolve the boundary $k[\ldots]$, giving rise to

$$n[\ R_1 \mid R_2 \mid P \mid m[\ldots]\ ]$$

Alternatively the sub-ambient m may exercise its capability to move outside n, $\mathtt{out}\langle n\rangle$, in which case the system will have three concurrent ambients:

$$k[\ldots] \quad | \quad n[\,\mathtt{open}\langle k\rangle.P\,] \quad | \quad m[\,Q_1 \mid Q_2\,]$$

Papers such as [6, 4] demonstrate that this calculus is very effective in formally describing the run-time behaviour of mobile agents. However we believe that the development of semantic theories for ambients has had more limited success. For example in [6] it is argued that the process

$$\boldsymbol{\nu} n\ n[P],$$

where n does not occur in P, can not be distinguished from the trivial process $\mathbf{0}$; intuitively the name n is unknown both inside and outside the ambient and consequently no other ambient can exercise a capability over it. This leads to the so-called *perfect firewall equation*

$$\boldsymbol{\nu} n\ n[P] \approx \mathbf{0}, \quad \text{for } n \text{ not in } P.$$

This raises two questions:

- What is the appropriate notion of semantic equivalence $\approx$ for ambients?
- What proof methods exist for establishing such equivalences?

Bisimulation relations, in their various forms, have proved to be very popular as a basis for semantic equivalences for a variety of process calculi, such as CCS, [10], and the Picalculus, [12]. Essentially the behaviour of processes is characterised using co-inductive relations defined over a *labelled transition system*, or *lts*, a collection of relations of the form

$$P \xrightarrow{\alpha} Q$$

Intuitively this means that the system P may perform the action α, typically by interacting with its environment or context, and be thereby transformed into the system Q. The co-inductive nature of these equivalences ensures that there are powerful proof techniques available for establishing identities, [14, 18, 15], addressing the second question above. Arguing for their appropriateness, the first question, is usually carried out by *contextual reasoning*. Intuitively the actions α in the judgement $P \xrightarrow{\alpha} Q$ represent some small context with which P can interact; more importantly it is shown that this collection of small contexts, codified as actions, are sufficient to capture all possible interactions which processes can have with

arbitrary contexts. In short the bisimulation relation over the lts characterises some naturally defined contextually defined behavioural equivalence, [14, 1]. This is the topic of the current paper:

> Can we define an lts based operational semantics for ambients, and an associated bisimulation equivalence, which can be justified contextually ?

In [9] it has been argued that the calculus MA, as given in, for example [6], is qualitatively different from more standard process calculi such as the Picalculus[12]. It is difficult for ambients to control potential interference from other ambients in their environment. For example ambients are always under the threat of being entered by an arbitrary ambient in its environment, and they have no means to forbid such actions if they so wish. To armour ambients with the means to protect themselves, if necessary, from the influence of their environment the authors add *co-capabilities*, for each of the standard ambient capabilities; this idea of every action having a co-action is borrowed from process calculi such as CCS or the Picalculus. Thus, for example, an ambient may now only exercise the capability $\mathtt{in}\langle n\rangle$, if the ambient n is also willing to exercise the corresponding co-capability $\overline{\mathtt{in}}\langle n\rangle$. In

$$m[\,\mathtt{in}\langle n\rangle.Q_1 \mid Q_2\,] \quad | \quad n[P]$$

the ambient m can migrate inside n if P has the form $\overline{\mathtt{in}}\langle n\rangle.P_1 \mid P_2$, in which case the system evolves to

$$n[\,m[\,Q_1 \mid Q_2\,] \mid P_1 \mid P_2\,]$$

That is m may only enter n if n allows it. The resulting calculus, called *Safe Ambients*, abbreviated SA, is shown to have a much more satisfactory equational theory, and numerous equations, often type dependent, may be found in [9]. Nevertheless these equations are expressed relative to a contextually defined equivalence. Establishing them requires, for the most part, reasoning about the effect arbitrary contexts may have on ambients.

We extend the syntax of ambients even further, by allowing capabilities to be defined relative to *passwords*. Co-capabilities give a certain amount of control to ambients over the ability of others to exercise capabilities on them; $\mathtt{in}\langle n\rangle$ can only be exercised if n is also willing to perform $\overline{\mathtt{in}}\langle n\rangle$. However n has no control over who obtains the capability $\mathtt{in}\langle n\rangle$. But if we generalise capabilities (and co-capabilities) to contain an extra component then this extra component may be used by n to exercise control over, and differentiate between, different ambients who may wish to exercise a capability. Now an ambient wishing to migrate inside n must exercise a capability of the form $\mathtt{in}\langle n, h\rangle$, for some password h; but the capability will

only have an effect if n exercises the corresponding co-capability, with the *same* password, $\overline{\mathtt{in}}\langle n, h\rangle$. By managing passwords, for example generating new ones and distributing them selectively, n may now program who may migrate into its computation space, and when. Moreover an ambient may provide different services depending on the passwords exhibited by its clients. We call this extended language *Safe Ambients with Passwords*, abbreviated SAP. It is formally defined, with a reduction semantics in Section 2.

Following the ideas of [8, 13] it is straightforward to define a contextual equivalence between terms in SAP, or indeed any of the many other variants of ambients. We let $\cong$ be the largest equivalence relation between terms which

- is a congruence for the language, that is is preserved by all constructs of the language
- preserves, in some sense, the reduction semantics of the language
- preserves *barbs*, that is preserves some simple observational property of terms.

A formal definition is given in Definition2.2; the only real parameter here is in the precise definition of the allowed *barbs*. As we shall see, in our setting the resulting equivalence is invariant with respect to a wide variety of possible barbs. This emphasises our opinion that the resulting equivalence $\cong$ is a reasonable semantic equivalence for SAP; by definition it has all of the extensional properties we require of such a relation.

The main result of the paper is

- an lts based operational semantics for SAP
- a bisimulation based equivalence over this lts, denoted $\approx$, which coincides with $\cong$.

In principle this opens up the theory of ambients, or at least those definable in SAP, to the co-inductive proof techniques associated with bisimulations. However there is a catch. The lts on which the operational semantics is based is far from trivial. As expected, it contains actions for all of the capabilities and co-capabilities in the language (here, as in much of the paper, we will ignore passwords unless they play a central role in the discussion). These take the form $P \stackrel{\alpha}{\longmapsto} Q$, a typical example being

$$\mathtt{in}\langle n\rangle.P \stackrel{\mathtt{in}\langle n\rangle}{\longmapsto} P$$

These actions do not prescribe any direct behaviour to individual ambients although they indirectly induce behaviour for particular ambients. For

example, the ambient

$$m[\,\mathtt{in}\langle n\rangle.P\,]$$

now has the ability to *enter* an ambient named n, because its body has the capability to perform the action $\mathtt{in}\langle n\rangle$. So our lts will also require actions of the form $\mathtt{enter}\langle n\rangle$, whose effects are in general higher-order. When such an action is performed we must prescribe

- which ambient enters n
- what residual code remains behind.

For example in the system

$$m[\,\mathtt{in}\langle n\rangle.P\,] \;\mid\; Q$$

if the action $\mathtt{enter}\langle n\rangle$ is performed the migrating ambient is $m[P]$, while the residual code is Q. In general the migrating ambient and the residual code may share private names and therefore to formalise actions such as $\mathtt{enter}\langle n\rangle$ we use a kind of *concretion*, [11, 16, 9]. Such actions will have the form

$$P \stackrel{\mathtt{enter}\langle n\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{m}\langle A\rangle_n P'$$

Here A is the migrating ambient, n the target, P' is the residual code and $\tilde{m}$ the shared names.
The details, including a formal definition of the higher-order lts, are given in Section 3. It includes a definition of internal actions, $\stackrel{\tau}{\longmapsto}$, which consists of one component of the system allowing the migration of an ambient, and another its reception; ambients are communicated between components in a system. One such definition, $\approx_{\mathrm{d}}$ called *delay bisimulation equivalence*, is given in Section 4.2, where as in [16, 22], agents are immediately tested after a visible action. In addition to being higher-order $\approx_{\mathrm{d}}$ is in *early* form, as instantiation takes place before residuals are compared. It is also in *weak* form, in that single actions $\stackrel{\alpha}{\longmapsto}$ are allowed to be matched by weak actions, $\stackrel{\alpha}{\Longmapsto}$. However because of the presence of concretions it is natural to define these actions so that internal transitions are only allowed before the action in question, α:

$$P \stackrel{\mathtt{enter}\langle n\rangle}{\Longmapsto} \boldsymbol{\nu}\tilde{m}\langle A\rangle_n P' \;\text{ if }\; P \stackrel{\tau}{\longmapsto}{}^{*} \stackrel{\mathtt{enter}\langle n\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{m}\langle A\rangle_n P'$$

In Section 4.2 we show that *delay bisimulation equivalence* is contained in the contextual congruence $\cong$ but in general does not coincide with it. Intuitively, in the definition of weak actions we need to allow internal actions to continue after the action has taken place; we need to allow communications between the components of a concretion. To incorporate

TABLE 1 The Calculus SAP

Names: $n, h, \ldots \in \mathbf{N}$

Processes:

$P ::=$	$\mathbf{0}$	nil process
	$P_1 \mid P_2$	parallel composition
	$\boldsymbol{\nu} n P$	restriction
	$C.P$	prefixing
	$n[P]$	ambient
	$!C.P$	replication

Capabilities:

$C ::=$	$\mathtt{in}\langle n, h\rangle$	may enter into n
	$\mathtt{out}\langle n, h\rangle$	may exit out of n
	$\mathtt{open}\langle n, h\rangle$	may open n
	$\overline{\mathtt{in}}\langle n, h\rangle$	allow enter
	$\overline{\mathtt{out}}\langle n, h\rangle$	allow exit
	$\overline{\mathtt{open}}\langle n, h\rangle$	allow open

this idea we define a new lts where now all actions take the form $P \xrightarrow{\alpha} Q$, that is all the residuals are processes; essentially concretions are eliminated by *applying* them to the processes which previously formed part of our definition of higher-order bisimulation. The main result of the paper is that, in SAP, the resulting (weak) bisimulation equivalence $\approx$ coincides with $\cong$.

Most of the paper uses a *pure* form of ambients, without any communication. In Section 5 we show that our results extend to a calculus in which messages can be sent and received within ambients, similarly to [6, 9]. In the following section we give some examples which indicate that our form of bisimulation may play a useful role in reasoning about ambient behaviour.

The paper ends with Section 7, containing a discussion of our results and a comparison with related work.

2 The Calculus SAP

The syntax of processes is given in Table 1 and is basically the same as that in [6], except that each of the original capabilities has a co-capability, as in [9], and that now each capability has an extra argument h, which

TABLE 2 Structural Congruence

$P \mid Q \equiv Q \mid P$	(Struct Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)
$P \mid \mathbf{0} \equiv P$	(Struct Zero Par)
$\boldsymbol{\nu} n\mathbf{0} \equiv \mathbf{0}$	(Struct Zero Res)
$\boldsymbol{\nu} n \boldsymbol{\nu} m P \equiv \boldsymbol{\nu} m \boldsymbol{\nu} n P$	(Struct Res Res)
$n \notin \mathrm{fn}(P)$ implies $\boldsymbol{\nu} n(P \mid Q) \equiv P \mid \boldsymbol{\nu} n Q$	(Struct Res Par)
$n \neq m$ implies $\boldsymbol{\nu} n(m[P]) \equiv m[\boldsymbol{\nu} n P]$	(Struct Res Amb)

may be looked upon as a *password*. Note also that we have replicated prefixing, rather than full replication, or recursion. Finally for simplicity we have omitted communication; this will be added in Section 5.

When writing examples we will use the standard conventions for ambients; trailing occurrences of $\mathbf{0}$ are omitted, $n[\mathbf{0}]$ will be shortened to $n[]$ and as usual parallel composition has the lowest precedence among all operators. We will also frequently write $\mathtt{in}\langle n \rangle$ to denote $\mathtt{in}\langle n, n \rangle$ and similarly for the other capabilities; in other words we will often use the name of an ambient as a password. The operator $\boldsymbol{\nu} n$ is a binder for names, leading to the usual notions of free and bound occurrences of names, $\mathrm{fn}(\cdot)$ and $\mathrm{bn}(\cdot)$, and α-conversion, $\equiv_\alpha$. We will identify processes up to α-conversion. More formally we will view process terms as representatives of their equivalence class with respect to $\equiv_\alpha$, and these representatives will always be chosen so that bound names are distinct from free names.

An equivalence relation $\mathcal{R}$ over processes is said to be a *congruence*, or *contextual*, if it is preserved by all the operators in the language. Formally this means it must satisfy the rules:

$P \mathcal{R} Q$ implies $\boldsymbol{\nu} n P \mathcal{R} \boldsymbol{\nu} n Q$	(Res)
$P \mathcal{R} Q$ implies $P \mid R \mathcal{R} Q \mid R$	(Par)
$P \mathcal{R} Q$ implies $n[P] \mathcal{R} n[Q]$	(Amb)
$P \mathcal{R} Q$ implies $C.P \mathcal{R} C.Q$	(Prefix)
$P \mathcal{R} Q$ implies $!P \mathcal{R} !Q$	(Repl)

We will say it is *p-contextual*, or partially contextual, if it is preserved by the structural operators, that is it satisfies all but the last two of these rules. *Structural congruence*, $\equiv$, is a *p-contextual* equivalence between processes, relating terms which we believe no reasonable semantics should distinguish. We define it to be the least congruence which satisfies the axioms and rules in Table 2. This will form a central role in the reduction semantics of our language, which we now explain.

This is given in terms of a binary relation over processes, $P \rightarrow Q$,

TABLE 3 Reduction Rules

$n[\mathtt{in}\langle m,h\rangle.P \mid Q] \mid m[\overline{\mathtt{in}}\langle m,h\rangle.R \mid S] \to m[\, n[P \mid Q] \mid R \mid S\,]$	(Red In)
$m[\, n[\mathtt{out}\langle m,h\rangle.P \mid Q] \mid R\,] \mid \overline{\mathtt{out}}\langle m,h\rangle.S \to n[P \mid Q] \mid m[R] \mid S$	(Red Out)
$\mathtt{open}\langle n,h\rangle.P \mid n[\,\overline{\mathtt{open}}\langle n,h\rangle.Q \mid R\,] \to P \mid Q \mid R$	(Red Open)
$P \equiv Q \quad Q \to R \quad R \equiv S$ implies $P \to S$	(Red Str)

which intuitively means that P can evolve to Q in one computation step. It is defined to be the least p-contextual relation which satisfies the axioms and rule in Figure 3. The axiom (Red In) describes how an ambient n may migrate *into* an ambient m. It must exercise the capability $\mathtt{in}\langle m,h\rangle$ for some password h, and at the same time m, the target computation space, must be willing to allow immigration, exercising the co-capability, $\overline{\mathtt{in}}\langle m,h\rangle$; note that the password is the same. Emigration *from* an ambient is described in the rule (Red Out), and is similar. The ambient n may attempt to emigrate from ambient m by exercising the capability $\mathtt{out}\langle m,h\rangle$; but the target computation space must allow entry, by exercising the corresponding co-capability with the same password, $\overline{\mathtt{out}}\langle m,h\rangle$. Note that in [9] this co-capability is exercised by m rather than the target computation space; we feel that with our definition there is a clearer distinction between the role of an ambient in a reduction and the corresponding role of its environment. Finally the axiom (Red Open) describes the circumstances under which the ambient n can be opened. Again it requires the co-operation of n, exercising the co-capability $\overline{\mathtt{open}}\langle n,h\rangle$, for some password h, before the capability $\mathtt{open}\langle n,h\rangle$ has an effect. The single rule (Red Str) merely states that reductions are made modulo structural congruence. In the sequel we will use $\Rightarrow$ to denote the reflexive, transitive closure of $\to$.

We end this section with a definition of what we believe to be a reasonable notion of behavioural equivalence. Following [13, 8] we simply say that $\cong$ is the *largest* equivalence between processes which satisfies some reasonable criteria. We require that

- it is a *congruence*, to aid in compositional verification,
- it is invariant, in some sense, with respect to the reduction relation $\to$. Formally we say a relation $\mathcal{R}$ is *reduction closed* if $P\,\mathcal{R}\,Q$ and $P \to P'$ implies the existence of some Q' such that $Q \Rightarrow Q'$ and $P'\,\mathcal{R}\,Q'$.
- it preserves some intuitive *observation predicate*, $P{\downarrow_n}$.

With ambients there are many ways of formulating observation predi-

cates. In [6] the predicate $P \downarrow_n$ is used to denote the possibility of process P of interacting with the environment via the ambient n; it is true whenever $P \equiv \boldsymbol{\nu}\tilde{m}(n[P_1] \mid P_2)$ where $n \notin \{\tilde{m}\}$. For MA, [6], this is a reasonable definition of observation as no authorisation is required to cross a boundary. As a consequence, the presence of an ambient n at top level denotes a potential interaction between the process and the environment via n. However in SA, [9], and our language SAP, the process $\boldsymbol{\nu}\tilde{m}(n[P_1] \mid P_2)$ only represents a potential interaction if P_1 can exercise an appropriate co-capability. For example in [9] $P\downarrow_n$ is defined to be true whenever $P \equiv \boldsymbol{\nu}\tilde{m}(n[C.P_1 \mid P_2] \mid P_3)$ where and $C \in \{\overline{\mathtt{in}}\langle n\rangle, \overline{\mathtt{open}}\langle n\rangle\}$ and $n \notin \{\tilde{m}\}$. We use a slight simplification of this definition.

Definition 2.1 (Barbs). *We write $P\downarrow_n$ if and only if there exist h, $\tilde{m}$, P_1, P_2, and P_3 such that*

$$P \equiv \boldsymbol{\nu}\tilde{m}(n[\overline{\mathtt{open}}\langle n, h\rangle.P_1 \mid P_2] \mid P_3)$$

where $n, h \notin \tilde{m}$. We write $P\Downarrow_n$ if $P \Rightarrow P'$ and $P'\downarrow_n$.

Note that the barb only mentions the ambient n and not the password used to open it; we could of course define a more detailed barb $P\downarrow_{n,h}$ but as we shall see this is unnecessary (see Theorem 4.4).

Definition 2.2 (Barbed Congruence). *Barbed congruence, written $\cong$, is the largest congruence relation over processes which*

- *is reduction closed*
- *barb preserving; that is $P \cong Q$ and $P\downarrow_n$ implies $Q\Downarrow_n$*

Our choice of observation here may feel arbitrary. But in Section 4 we will show that $\cong$ remains invariant under a large choice of possible observation predicates.

Our aim is to give a co-inductive characterisation of $\cong$ using an lts based operational semantics for SAP.

3 A Labelled Transition Semantics

The capabilities or prefixes C in our language give rise, in the standard manner, [10], to actions of the form $P \stackrel{C}{\longmapsto} Q$; for example we would have

$$\overline{\mathtt{in}}\langle n, h\rangle.P_1 \mid P_2 \stackrel{\overline{\mathtt{in}}\langle n,h\rangle}{\longmapsto} P_1 \mid P_2$$

These actions could be used to define a versions of weak bisimulation equivalence over processes, $\approx_{bad}$, again in the standard manner, [10]. However it should be obvious that $\approx_{bad}$ is unsatisfactory as a notion of equivalence for SAP. For example these actions cannot be performed by ambients

TABLE 4 Labels, Concretions, and Outcomes

Actions:	μ	::=	$\mathtt{in}\langle n,h\rangle$	$\mathtt{out}\langle n,h\rangle$	$\mathtt{open}\langle n,h\rangle$	
			$\overline{\mathtt{in}}\langle n,h\rangle$	$\overline{\mathtt{out}}\langle n,h\rangle$	$\overline{\mathtt{open}}\langle n,h\rangle$	
Labels:	α	::=	τ	μ	$\mathtt{enter}\langle n,h\rangle$	$\overline{\mathtt{enter}}\langle n,h\rangle$
			$\mathtt{exit}\langle n,h\rangle$	$\mathtt{pop}\langle n,h\rangle$	$\mathtt{free}\langle n,h\rangle$	
Concretions:	K	::=	$\boldsymbol{\nu}\tilde{m}\langle P\rangle_n Q$			
Outcomes:	O	::=	P	K		

and therefore we would have the identity

$$n[P] \approx_{bad} \mathbf{0}$$

regardless of P.

However the actions above can be considered the basis of further capabilities. For example in the system $n[\mathtt{in}\langle m\rangle P.] \mid Q$ there is the capability to *enter* ambient m. Exercising this capability has a dual effect; on the one hand the ambient $n[P]$ will actually move into the ambient m, on the other the process Q will remain executing at the point at which the capability is exercised. In general each of the simple prefix actions C will induce different, more complicated capabilities in ambients, and more generally processes. These will be formulated as actions of the form

$$P \stackrel{\alpha}{\longmapsto} O$$

where the range of α and of O, the *outcomes*, are given in Table 4. These outcomes may be a simple process Q, if for example α is a prefix from the language, or a *concretion*, of the form

$$\boldsymbol{\nu}\tilde{m}\langle P\rangle_n Q$$

Here, intuitively, process P represents what must stay inside an ambient n whereas process Q must stay outside n, and $\tilde{m}$ is the set of private names shared by P and Q.

The rules defining our labelled transition semantics, inspired by [9, 3], are given in Table 5 and Table 6 and are in *late* style [12]. Let us first examine those induced by the prefix $\mathtt{in}$, the *immigration* of ambients. Here we will ignore the use of passwords as they play no role in our explanations. A typical example of an ambient m migrating into an ambient n is as

TABLE 5 Labelled Transition System - Enter and Exit

$$\text{(Enter)}\ \frac{P \overset{\mathtt{in}\langle n,h\rangle}{\longmapsto} P'}{m[P] \overset{\mathtt{enter}\langle n,h\rangle}{\longmapsto} \langle m[P']\rangle_n \mathbf{0}} \qquad \text{(Co-Enter)}\ \frac{P \overset{\overline{\mathtt{in}}\langle n,h\rangle}{\longmapsto} P'}{n[P] \overset{\overline{\mathtt{enter}}\langle n,h\rangle}{\longmapsto} \langle P'\rangle_n \mathbf{0}}$$

$$(\tau\ \text{In})\ \frac{P \overset{\mathtt{enter}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{p}\langle P_1\rangle_n P_2 \qquad Q \overset{\overline{\mathtt{enter}}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{q}\langle Q_1\rangle_n Q_2}{P \mid Q \overset{\tau}{\longmapsto} \boldsymbol{\nu}\tilde{p}\boldsymbol{\nu}\tilde{q}(n[P_1 \mid Q_1] \mid P_2 \mid Q_2)}$$

$$\text{if}\ \ ((\mathrm{fn}(P_1) \cup \mathrm{fn}(P_2)) \cap \{\tilde{q}\}) = ((\mathrm{fn}(Q_1) \cup \mathrm{fn}(Q_2)) \cap \{\tilde{p}\}) = \emptyset$$

$$\text{(Exit)}\ \frac{P \overset{\mathtt{out}\langle n,h\rangle}{\longmapsto} P'}{m[P] \overset{\mathtt{exit}\langle n,h\rangle}{\longmapsto} \langle \mathbf{0}\rangle_n m[P']} \qquad \text{(Pop)}\ \frac{P \overset{\mathtt{exit}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{m}\langle P_1\rangle_n P_2}{n[P] \overset{\mathtt{pop}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{m}(n[P_1] \mid P_2)}$$

$$(\tau\ \text{Out})\ \frac{P \overset{\mathtt{pop}\langle n,h\rangle}{\longmapsto} P' \qquad Q \overset{\overline{\mathtt{out}}\langle n,h\rangle}{\longmapsto} Q'}{P \mid Q \overset{\tau}{\longmapsto} P' \mid Q'}$$

follows:

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

The driving force behind the migration is the activation of the prefix $\mathtt{in}\langle n\rangle$, within the ambient m. It induces a capability in the ambient m to migrate into n, which we formalise as a new action $\mathtt{enter}\langle n\rangle$. Thus an application of the rule (Enter) gives

$$m[\mathtt{in}\langle n\rangle.P] \overset{\mathtt{enter}\langle n\rangle}{\longmapsto} \langle m[P]\rangle_n \mathbf{0}$$

and more generally, using the structural rules

$$m[\mathtt{in}\langle n\rangle.P_1] \mid P_2 \overset{\mathtt{enter}\langle n\rangle}{\longmapsto} \langle m[P_1]\rangle_n P_2$$

This means that the system $m[\mathtt{in}\langle n\rangle.P_1] \mid P_2$ has the capability to enter an ambient n; if the capability is exercised the ambient $m[P_1]$ will enter n while P_2 will be the residual at the point of execution. Of course the action can only be executed if there is a corresponding co-action $\overline{\mathtt{enter}}\langle n\rangle$ performed by n. The rule (Co-Enter) allows these to be derived. So for example we have

$$n[\overline{\mathtt{in}}\langle n\rangle.Q_1] \mid Q_2 \overset{\overline{\mathtt{enter}}\langle n\rangle}{\longmapsto} \langle Q_1\rangle_n Q_2$$

Here, after the action, Q_1 remains inside n, while Q_2 is outside. The

communication (τ In) now allows these two complementary actions to occur simultaneously, effecting the migration of the ambient $m[P_1]$ from its current computation space into the ambient n, giving rise to the original move above:

$$m[\mathtt{in}\langle n\rangle.P_1] \mid P_2 \mid n[\overline{\mathtt{in}}\langle n\rangle.Q_1] \mid Q_2 \stackrel{\tau}{\longmapsto} n[\, m[P_1] \mid Q_1\,] \mid P_2 \mid Q_2$$

Note that this is a *higher-order* interaction, as the ambient $m[P_1]$ is transferred between two computation spaces.

The structural rule (Res) in Table 6 allows the migrating ambient to share private names with its point of origin, in the same manner as in the Picalculus. This rule employs the convention that if O is the concretion $\boldsymbol{\nu}\tilde{m}\langle P\rangle_n Q$, then $\boldsymbol{\nu}rO$ is a shorthand for $\boldsymbol{\nu}\tilde{m}\langle P\rangle_n \boldsymbol{\nu}rQ$, if $r \notin \mathrm{fn}(P)$, and the concretion $\boldsymbol{\nu}(r\tilde{m})\langle P\rangle_n Q$ otherwise. We have a similar convention for the rule (Par): $O \mid R$ is defined to be the concretion $\boldsymbol{\nu}\tilde{m}\langle P\rangle_n(Q \mid R)$, where $\tilde{m}$ are chosen, using α-conversion if necessary, so that $\mathrm{fn}(R) \cap \{\tilde{m}\} = \emptyset$. So, for example if k occurs free in both P_1 and P_2 we have the action

$$\boldsymbol{\nu}k(m[\mathtt{in}\langle n\rangle.P_1] \mid P_2) \stackrel{\mathtt{enter}\langle n\rangle}{\longmapsto} \boldsymbol{\nu}k(\langle m[P_1]\rangle_n P_2)$$

and the rule (τ In) now gives

$$\boldsymbol{\nu}k(m[\mathtt{in}\langle n\rangle.P_1] \mid P_2) \mid n[\overline{\mathtt{in}}\langle n\rangle.Q_1] \mid Q_2 \stackrel{\tau}{\longmapsto} \boldsymbol{\nu}k(n[m[P_1] \mid Q_1] \mid P_2 \mid Q_2)$$

where it is assumed that k is chosen to be fresh to m, Q_1 and Q_2. So the resulting process is structurally equivalent to

$$\boldsymbol{\nu}k(n[\, m[P_1] \mid Q_1\,] \mid P_2) \mid Q_2$$

Note that the scope of k has now extended to include the ambient n.

The rules of *emigration* are organised in a similar manner, although they are slightly more complicated. A typical example of ambient m emigrating from ambient n is as follows:

$$n[\, m[\mathtt{out}\langle n\rangle.P_1] \mid P_2\,] \mid \overline{\mathtt{out}}\langle n\rangle.Q \;\rightarrow n[P_2] \mid m[P_1] \mid Q$$

The driving force behind the emigration is the activation of the prefix $\mathtt{out}\langle n\rangle$ within the ambient m; however its effect is even more indirect than that of the prefix $\mathtt{in}\langle n\rangle$. It induces a capability in the ambient m to emigrate from n, which we formalise as a new action $\mathtt{exit}\langle n\rangle$. Thus an application of the rule (Exit), followed by (Par Exit) gives

$$m[\,\mathtt{out}\langle n\rangle.P_1\,] \mid P_2 \stackrel{\mathtt{exit}\langle n\rangle}{\longmapsto} \langle P_2\rangle_n m[P_1]$$

Here when this capability is exercised the code P_2 will remain inside the ambient n while the ambient $m[P_1]$ will be outside. However to actually effect the emigration of m we need a further context, namely the ambient

TABLE 6 Labelled Transition System - Part 2

$$\text{(Act)}\ \frac{}{\mu.P \overset{\mu}{\longmapsto} P} \qquad \text{(Repl Act)}\ \frac{}{!\mu.P \overset{\mu}{\longmapsto} P \mid !\mu.P}$$

$$\text{(Free)}\ \frac{P \overset{\overline{\mathtt{open}}\langle n,h\rangle}{\longmapsto} P'}{n[P] \overset{\mathtt{free}\langle n,h\rangle}{\longmapsto} P'} \qquad (\tau\ \text{Open})\ \frac{P \overset{\mathtt{open}\langle n,h\rangle}{\longmapsto} P' \quad Q \overset{\mathtt{free}\langle n,h\rangle}{\longmapsto} Q'}{P \mid Q \overset{\tau}{\longmapsto} P' \mid Q'}$$

Structural Rules :

$$\text{(Par)}\ \frac{P \overset{\alpha}{\longmapsto} O \ \ \alpha \neq \mathtt{exit}\langle n,h\rangle}{P \mid Q \overset{\alpha}{\longmapsto} O \mid Q} \qquad \text{(Par Exit)}\ \frac{P \overset{\mathtt{exit}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{m}\langle P_1\rangle_n P_2}{P \mid Q \overset{\mathtt{exit}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{m}\langle P_1 \mid Q\rangle_n P_2}$$

$$\text{(Res)}\ \frac{P \overset{\alpha}{\longmapsto} O \qquad n \notin \mathrm{fn}(\alpha)}{\boldsymbol{\nu}nP \overset{\alpha}{\longmapsto} \boldsymbol{\nu}nO} \qquad (\tau\ \text{Amb})\ \frac{P \overset{\tau}{\longmapsto} Q}{n[P] \overset{\tau}{\longmapsto} n[Q]}$$

n from which to emigrate. This leads to another action, called $\mathtt{pop}\langle n\rangle$, with the associated rule (Pop); an application of which gives:

$$n[\, m[\mathtt{out}\langle n\rangle.P_1] \ \mid\ P_2\,] \overset{\mathtt{pop}\langle n\rangle}{\longmapsto} n[P_2] \ \mid\ m[P_1]$$

However this action, $\mathtt{pop}\langle n\rangle$, is only possible if the environment allows the emigration of ambient n, controlled by the co-action $\overline{\mathtt{out}}\langle n\rangle$, and codified in the rule (τ Out); an application of which gives the original move above:

$$n[\, m[\mathtt{out}\langle n\rangle.P_1] \mid P_2\,] \ \mid\ \overline{\mathtt{out}}\langle n\rangle.Q \overset{\tau}{\longmapsto} n[P_2] \ \mid\ m[P_1] \ \mid\ Q$$

Finally let us consider the rules which control the *opening* of ambients, which are considerably more straightforward. The opening of an ambient n is activated by the prefix $\mathtt{open}\langle n\rangle$ but it is controlled by ambient n via the prefix $\overline{\mathtt{open}}\langle n\rangle$. Thus an application of the rule (Free) gives

$$n[\,\overline{\mathtt{open}}\langle n\rangle.P\,] \overset{\mathtt{free}\langle n\rangle}{\longmapsto} P$$

and an application of the rule (τ Open)

$$n[\,\overline{\mathtt{open}}\langle n\rangle.P\,] \ \mid\ \mathtt{open}\langle n\rangle.Q \overset{\tau}{\longmapsto} P \ \mid\ Q$$

It is worth noting that, according to our semantics only ambients, rather then general code, can migrate:

LEMMA 3.1.

- *If* $P \stackrel{\texttt{enter}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{m}\langle P_1\rangle_n P_2$ *then* P_1 *is an ambient*
- *If* $P \stackrel{\texttt{exit}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{m}\langle P_1\rangle_n P_2$ *then* P_2 *is an ambient.*

Proof: By rule induction. □

We end this section with a theorem which asserts that the lts based semantics coincides with the reduction semantics of Section 2.

Theorem 3.2.

1. *If* $P \stackrel{\tau}{\longmapsto} P'$ *then* $P \to P'$.
2. *If* $P \to P'$ *then* $P \stackrel{\tau}{\longmapsto} \equiv P'$.

Proof: By transition induction. Part 1 is the most difficult. It requires a result describing the structure of a process P and the outcome O for any action α such that $P \stackrel{\alpha}{\longmapsto} O$. For instance,

- If $P \stackrel{\texttt{enter}\langle n,h\rangle}{\longmapsto} O$ then there exist $\tilde{p}, m, P_1, P_2, P_3$, with $n, h \notin \tilde{p}$, such that $P \equiv \boldsymbol{\nu}\tilde{p}(m[\texttt{in}\langle n,h\rangle.P_1 \mid P_2] \mid P_3)$ and $O \equiv \boldsymbol{\nu}\tilde{p}\langle m[P_1 \mid P_2]\rangle_n P_3$
- If $P \stackrel{\texttt{exit}\langle n,h\rangle}{\longmapsto} O$ then there exist $\tilde{p}, m, P_1, P_2, P_3$, with $n, h \notin \tilde{p}$, such that $P \equiv \boldsymbol{\nu}\tilde{p}(m[\texttt{out}\langle n,h\rangle.P_1 \mid P_2] \mid P_3)$. and $O \equiv \boldsymbol{\nu}\tilde{p}\langle P_3\rangle_n m[P_1 \mid P_2]$.

Similar sub-results are necessary for the remaining actions. The proof of these results is standard. □

Note that the proof above requires extending the definition of $\equiv$ to concretions.

4 The Characterisation

In the first subsection we re-examine our definition of barbed congruence, $\cong$, showing that it is very robust under changes to the precise definition of barbs. This is followed by our first attempt at a bisimulation equivalence, called *delay bisimilarity*; we show this is contained in $\cong$ but does not in general coincide with it. In the final section we modify *delay bisimilarity*, to give our co-inductive characterisation of $\cong$.

4.1 Barbs

As already mentioned in Section 2 according to [6, 9], the predicate $P{\downarrow_n}$ detects the ability of a process P to interact with its environment via the ambient n. However, in other process calculi, like the Picalculus, barbs are defined using (visible) actions. So, one may wonder how our definition of barbed congruence would be affected by inheriting the notion of barb from the lts. In fact we can show that our definition of barb coincides with the choice of a particular action:

LEMMA 4.1. *$P\downarrow_n$ iff $P \stackrel{\mathtt{free}\langle n,h\rangle}{\longmapsto} P'$ for some h and P'.*

Proof: Straightforward. □

In this subsection, we prove that for all possible labels generated in our lts the resulting definitions of barbed congruence collapse and coincide with $\cong$. We recall that α ranges over the labels defined in Table 4.

DEFINITION 4.2. *We write $P\downarrow_\alpha$ if $P \stackrel{\alpha}{\longmapsto}$. We write $P\Downarrow_\alpha$ if $P \Rightarrow \stackrel{\alpha}{\longmapsto}$.*

DEFINITION 4.3. *Let $\mathcal{L}$ denote the labels* $\mathtt{in}$, $\mathtt{out}$, $\mathtt{open}$, $\overline{\mathtt{in}}$, $\overline{\mathtt{out}}$, $\overline{\mathtt{open}}$, $\mathtt{enter}$, $\overline{\mathtt{enter}}$, $\mathtt{exit}$, $\mathtt{pop}$ *and* $\mathtt{free}$. *For each $\lambda \in \mathcal{L}$ let $\cong_\lambda$ be the largest congruence over processes which*

- *is* reduction closed
- *preserves λ barbs; that is $P \cong_\lambda Q$ and $P \downarrow_{\lambda\langle n,h\rangle}$ implies $Q \Downarrow_{\lambda\langle n,h\rangle}$.*

It is very easy to establish that if $P \cong_\lambda Q$ then

- $P\Downarrow_n$ iff $Q\Downarrow_n$
- $P \Rightarrow P'$ implies $Q \Rightarrow Q'$ for some Q' such that $P' \cong_\lambda Q'$.

In the sequel we will use these properties without comment.

THEOREM 4.4. *Let P and Q be two processes, then*

$$\forall\, \lambda \in \mathcal{L} \qquad P \cong Q \quad \text{iff} \quad P \cong_\lambda Q.$$

Proof: Since the definitions of $\cong$ and $\cong_\lambda$ differ only in the notion of barb it suffices to show that the two forms of barbs imply each other. We examine two examples of λ; the other cases are similar.

- $\lambda = \mathtt{pop}$.
  Let us consider first the implication from left to right. Let $P \cong Q$ and $P\downarrow_{\mathtt{pop}\langle n,h\rangle}$; we want to conclude that $Q\Downarrow_{\mathtt{pop}\langle n,h\rangle}$.
  Consider the context

  $$S_1[\cdot] \stackrel{\text{def}}{=} [\cdot] \mid \overline{\mathtt{out}}\langle n,h\rangle.f[\overline{\mathtt{open}}\langle f\rangle]$$

  It is easy to prove that whenever f is fresh to R,

  $$R\Downarrow_{\mathtt{pop}\langle n,h\rangle} \quad \text{iff} \quad S_1[R]\Downarrow_f$$

  This is sufficient to establish the result. For $P \cong Q$ implies $S_1[P] \cong S_1[Q]$ which in turn implies $S_1[Q]\Downarrow_f$, from which we have the required $Q\Downarrow_{\mathtt{pop}\langle n,h\rangle}$.
  As to the implication from right to left, let $P \cong_{\mathrm{pop}} Q$ and $P\downarrow_n$, then we want to conclude that $Q\Downarrow_n$. Again this involves using a context.

We recall, that by Lemma 4.1, if $P\downarrow_n$ then there exists h such that $P \stackrel{\mathtt{free}\langle n,h\rangle}{\longmapsto}$. Thus, we define a context:

$$S_2^h[\cdot] \stackrel{\text{def}}{=} [\cdot] \mid \mathtt{open}\langle n,h\rangle.k[r[\overline{\mathtt{out}}\langle k\rangle]]$$

This is constructed so that whenever r and k are fresh to R then:

- $S_2^h[R]\Downarrow_{\mathtt{pop}\langle k\rangle}$ implies $R\Downarrow_n$
- $R\Downarrow_n$ implies $\exists h.\ S_2^h[R]\Downarrow_{\mathtt{pop}\langle k\rangle}$.

This is sufficient to establish $Q\Downarrow_n$.

- $\lambda = \overline{\mathtt{enter}}$.
  Again this is a question of defining two appropriate contexts. Let

$$S_1[\cdot] \stackrel{\text{def}}{=} [\cdot] \mid f[\mathtt{in}\langle n,h\rangle.\mathtt{out}\langle n,k\rangle] \mid \overline{\mathtt{out}}\langle n,k\rangle.g[\overline{\mathtt{open}}\langle g\rangle]$$

This context has the property that

$$R\Downarrow_{\overline{\mathtt{enter}}\langle n,h\rangle} \quad \text{iff} \quad S_1[R]\Downarrow_g$$

whenever f, g and k are fresh to R. For the reverse direction we let

$$S_2^h[\cdot] \stackrel{\text{def}}{=} [\cdot] \mid \mathtt{open}\langle n,h\rangle.g[\overline{\mathtt{in}}\langle g\rangle]$$

This has the required property that:

- $S_2^h[R]\Downarrow_{\overline{\mathtt{enter}}\langle g\rangle}$ implies $R\Downarrow_n$
- $R\Downarrow_n$ implies $\exists h.\ S_2^h[R]\Downarrow_{\overline{\mathtt{enter}}\langle g\rangle}$.

provided that g is fresh to R.

□

As expected, the use of passwords is fundamental to the above result. In particular in the the case $\lambda = \overline{\mathtt{enter}}$ the use of the fresh password k in the definition of $S_1[\cdot]$ is essential. Note also that this case, $\lambda = \overline{\mathtt{enter}}$, shows that the Levi and Sangiorgi's definition of barb, [9], can be simplified, to coincide with our original definition.

4.2 Delay Bisimilarity

Since we are interested in weak bisimilarities we have to define the notion of a weak action. Inspired by Sangiorgi's bisimilarities for HOπ [16] we first focus on a form of *delay bisimilarity* where agents are immediately tested after a visible action.

Definition 4.5.

- $\stackrel{\alpha}{\Longmapsto}$ *denotes* $\stackrel{\tau}{\longmapsto}^* \stackrel{\alpha}{\longmapsto}$

- $\stackrel{\hat{\alpha}}{\Longmapsto}$ *denotes* $\stackrel{\tau}{\longmapsto}^*$ *if* $\alpha = \tau$ *and* $\stackrel{\alpha}{\Longmapsto}$ *otherwise.*

Since some actions result in concretions rather than processes we need to have a method of comparing concretions. This will be carried out implicitly by applying them to processes.

DEFINITION 4.6. $\boldsymbol{\nu}\tilde{m}\langle P\rangle_n Q \bullet R \stackrel{\text{def}}{=} \boldsymbol{\nu}\tilde{m}(n[P \mid R] \mid Q)$ *where* $\{\tilde{m}\}$ *is chosen so that* $\{\tilde{m}\} \cap \text{fn}(R) = \emptyset$.

DEFINITION 4.7 (DELAY BISIMILARITY). *A symmetric relation* $\mathcal{S}$ *over processes is a* delay bisimulation *if* $P \ \mathcal{S} \ Q$ *implies:*

- *If* $P \stackrel{\alpha}{\longmapsto} P'$ *then there exists* Q' *such that* $Q \stackrel{\hat{\alpha}}{\Longmapsto} Q'$ *and* $P' \ \mathcal{S} \ Q'$.
- *If* $P \stackrel{\alpha}{\longmapsto} K_1$, $\alpha \neq \overline{\texttt{enter}}\langle n, h\rangle$, *then for all* R *there exists* K_2 *such that* $Q \stackrel{\alpha}{\Longmapsto} K_2$ *and* $K_1 \bullet R \ \mathcal{S} \ K_2 \bullet R$.
- *If* $P \stackrel{\overline{\texttt{enter}}\langle n,h\rangle}{\longmapsto} K_1$ *then for all* m *and* R *there exists* K_2 *such that* $Q \stackrel{\overline{\texttt{enter}}\langle n,h\rangle}{\Longmapsto} K_2$ *and* $K_1 \bullet m[R] \ \mathcal{S} \ K_2 \bullet m[R]$.

P and Q are delay bisimilar, written $P \approx_{\text{d}} Q$, *if* $P \ \mathcal{S} \ Q$ *for some delay bisimulation* $\mathcal{S}$.

Note that the $\stackrel{\overline{\texttt{enter}}\langle n,h\rangle}{\longmapsto}$ clause is treated separately because, as stated in Lemma 3.1, only ambients (rather than arbitrary code R) can enter ambient n.

THEOREM 4.8. *Delay bisimilarity is a congruence.*

Proof: It is straightforward to prove that $\approx_{\text{d}}$ is preserved by prefixing and iteration. We treat the other three constructs simultaneously.
Let $\mathcal{S}$ be the least equivalence relation such that:

- $\approx_{\text{d}} \subseteq \mathcal{S}$
- $P \ \mathcal{S} \ Q$ implies $P \mid R \ \mathcal{S} \ Q \mid R$ for all processes R
- $P \ \mathcal{S} \ Q$ implies $n[P] \ \mathcal{S} \ n[Q]$
- $P \ \mathcal{S} \ Q$ implies $\boldsymbol{\nu}nP \ \mathcal{S} \ \boldsymbol{\nu}nQ$

We prove that $\mathcal{S}$ is a delay bisimilarity up to $\equiv$, by induction on why two processes P and Q are in $\mathcal{S}$. The case when $P \approx_{\text{d}} Q$ follows by definition.

- $P \mid R \ \mathcal{S} \ Q \mid R$ because $P \ \mathcal{S} \ Q$. We now do induction on why $P \mid R \stackrel{\alpha}{\longmapsto} O$. Most cases are straightforward. We focus on the most interesting ones.

– $P \mid R \stackrel{\tau}{\longmapsto} O$ because $P \stackrel{\mathtt{enter}\langle n,h\rangle}{\longmapsto} K_1$ and $R \stackrel{\overline{\mathtt{enter}}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{r}\langle R_1\rangle_n R_2$, where O is structurally equivalent to $\boldsymbol{\nu}\tilde{r}((K_1 \bullet R_1) \mid R_2)$. We must find a matching move $Q \mid R \stackrel{\tau}{\longmapsto} O'$ such that $O \;\mathcal{S}\; O'$.
As $P \;\mathcal{S}\; Q$, we may use induction to find an action $Q \stackrel{\mathtt{enter}\langle n,h\rangle}{\Longmapsto} K_2$ such that $K_1 \bullet R_1 \;\mathcal{S}\; K_2 \bullet R_1$. Using the rule ($\tau$ In)we then have $Q \mid R \Longrightarrow \boldsymbol{\nu}\tilde{r}((K_2 \bullet R_1) \mid R_2)$ and since $\mathcal{S}$ is preserved by parallel composition and restriction, it follows that the required O' is $\boldsymbol{\nu}\tilde{r}((K_2 \bullet R_1) \mid R_2)$.

– $P \mid R \stackrel{\tau}{\longmapsto} O$ because $P \stackrel{\overline{\mathtt{enter}}\langle n,h\rangle}{\longmapsto} K_1$ and $R \stackrel{\mathtt{enter}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{r}\langle R_1\rangle_n R_2$, where again we can take O to be structurally equivalent to $\boldsymbol{\nu}\tilde{r}((K_1 \bullet R_1) \mid R_2)$. Note however that from Lemma 3.1 we can assume R_1 is an ambient. So here we can still use the induction hypothesis, $P \;\mathcal{S}\; Q$, to find a move $Q \stackrel{\overline{\mathtt{enter}}\langle n,h\rangle}{\Longmapsto} K_2$ such that $K_1 \bullet R_1 \;\mathcal{S}\; K_2 \bullet R_1$. So again the rule ($\tau$ In) can be used to deduce the required corresponding move from $Q \mid R$, namely $Q \mid R \Longmapsto \boldsymbol{\nu}\tilde{r}((K_1 \bullet R_1) \mid R_2)$.

– $P \mid R \stackrel{\mathtt{exit}\langle n,h\rangle}{\longmapsto} O$ because $P \stackrel{\mathtt{exit}\langle n,h\rangle}{\longmapsto} K_1 = \boldsymbol{\nu}\tilde{p}\langle P_1\rangle_n P_2$, and O is structurally equivalent to $\boldsymbol{\nu}\tilde{p}\langle P_1 \mid R\rangle_n P_2$, This case deserves to be treated separately due to the unusual rule (Par Exit) in Table 6. We must show that for an arbitrary process S there is a move $Q \mid R \Longrightarrow K_2$ such that $\boldsymbol{\nu}\tilde{p}\langle P_1 \mid R \mid S\rangle_n P_2 \;\mathcal{S}\; K_2 \bullet S$. Note that $\boldsymbol{\nu}\tilde{p}\langle P_1 \mid R \mid S\rangle_n P_2$ is structurally equivalent to $K_1 \bullet (R \mid S)$.
As $P \;\mathcal{S}\; Q$, by induction we know that for any process W there exists a concretion $K_3 = \boldsymbol{\nu}\tilde{q}\langle Q_1\rangle_n Q_2$ such that $Q \stackrel{\mathtt{exit}\langle n,h\rangle}{\Longmapsto} K_3$ and $K_1 \bullet W \;\mathcal{S}\; K_3 \bullet W$. In particular let us choose the concretion obtained when W is the process $R \mid S$. In this case we know $K_1 \bullet (R \mid S) \;\mathcal{S}\; K_3 \bullet (R \mid S)$.
The required concretion K_2 is then $\boldsymbol{\nu}\tilde{q}\langle Q_1 \mid R\rangle_n Q_2$ since $K_3 \bullet (R \mid S)$ is structurally equivalent to $K_2 \bullet S$ and an application of (Par Exit) gives $Q \mid R \Longrightarrow K_2$.

– The other cases are straightforward and are left to the reader.

- $n[P] \;\mathcal{S}\; n[Q]$ because $P \;\mathcal{S}\; Q$. We do induction on why $n[P] \stackrel{\alpha}{\longmapsto} O$. We give details only in the cases when α is $\mathtt{enter}\langle m,h\rangle$, or $\mathtt{pop}\langle n,h\rangle\}$. The remaining cases, when α is τ, $\overline{\mathtt{enter}}\langle n,h\rangle$, $\mathtt{exit}\langle m,h\rangle$, or $\mathtt{free}\langle n,h\rangle\}$, are similar.

– Let $n[P] \stackrel{\mathtt{enter}\langle m,h\rangle}{\longmapsto} K_1 = \langle P'\rangle_n \mathbf{0}$ because $P \stackrel{\mathtt{in}\langle m,h\rangle}{\longmapsto} P'$. For an

arbitrary process W we are required to find a move $n[Q] \stackrel{\mathtt{enter}\langle m,h\rangle}{\longmapsto} K_2$ such that $K_1 \bullet W \;\; \mathcal{S} \;\; K_2 \bullet W$.

As $P \;\; \mathcal{S} \;\; Q$, we may use induction to find a Q' such that $Q \stackrel{\mathtt{in}\langle m,h\rangle}{\longmapsto} Q'$ and $P' \;\; \mathcal{S} \;\; Q'$. We may now let $K_2 = \langle Q' \rangle_n \mathbf{0}$ as $n[Q] \stackrel{\mathtt{enter}\langle m,h\rangle}{\Longmapsto} K_2$ and since $\mathcal{S}$ is preserved by parallel composition and ambient constructor, we get $K_1 \bullet W = n[P' \mid W] \;\; \mathcal{S} \;\; n[Q' \mid W] = K_2 \bullet W$, as desired.

- Let $n[P] \stackrel{\mathtt{pop}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{p}(n[P_1] \mid P_2)$ because $P \stackrel{\mathtt{exit}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{p}\langle P_1 \rangle_n P_2 = K_1$. As $P \;\; \mathcal{S} \;\; Q$, by induction it holds that for all processes W there exists $K_2 = \boldsymbol{\nu}\tilde{q}\langle Q_1 \rangle_n Q_2$ such that $Q \stackrel{\mathtt{exit}\langle n,h\rangle}{\Longmapsto} K_2$ and $K_1 \bullet W \;\; \mathcal{S} \;\; K_2 \bullet W$. So, choosing the particular case when W is $\mathbf{0}$ we have $n[Q] \stackrel{\mathtt{pop}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{q}(n[Q_1] \mid Q_2)$ and $\boldsymbol{\nu}\tilde{p}(n[P_1] \mid P_2) \equiv K_1 \bullet \mathbf{0} \;\; \mathcal{S} \;\; K_2 \bullet \mathbf{0} \equiv \boldsymbol{\nu}\tilde{q}(n[Q_1] \mid Q_2)$, as desired.
- The inductive cases, for both kinds of actions, are straightforward.

- The remaining case, when $\boldsymbol{\nu}nP \;\; \mathcal{S} \;\; \boldsymbol{\nu}nQ$ because $P \;\; \mathcal{S} \;\; Q$, is straightforward and left to the reader.

□

In the sequel of the paper it will be useful to have a form of *internal choice.*

Definition 4.9. *Given any processes P and Q we have*

$$P \oplus Q \stackrel{\text{def}}{=} \boldsymbol{\nu}r(\mathtt{open}\langle r\rangle.P \mid \mathtt{open}\langle r\rangle.Q \mid r[\overline{\mathtt{open}}\langle r\rangle])$$

with $r \notin \text{fn}(P,Q)$.

Note that up to structural congruence $P \oplus Q \stackrel{\tau}{\longrightarrow} P \mid \boldsymbol{\nu}r(\mathtt{open}\langle r\rangle.Q)$ and $P \oplus Q \stackrel{\tau}{\longrightarrow} Q \mid \boldsymbol{\nu}r(\mathtt{open}\langle r\rangle.P)$. However for virtually any behavioural equivalence we will have $\boldsymbol{\nu}r(\mathtt{open}\langle r\rangle.R) = \mathbf{0}$ and so for the sake of simplicity we will simply write $P \oplus Q \stackrel{\tau}{\longrightarrow} P$ and $P \oplus Q \stackrel{\tau}{\longrightarrow} P$, when analysing computations involving $\oplus$.

Proposition 4.10. *Delay bisimilarity is strictly contained in barbed congruence.*

Proof: By Theorem 4.8 and Lemma 4.1 delay bisimilarity is contained in barbed congruence. To prove that delay bisimilarity is strictly contained in barbed congruence considers the following law:

$$!\mathtt{in}\langle a\rangle.(\mathtt{in}\langle b\rangle \oplus \mathtt{in}\langle c\rangle) = !\mathtt{in}\langle a\rangle.(\mathtt{in}\langle b\rangle \oplus \mathtt{in}\langle c\rangle) \mid \mathtt{in}\langle a\rangle.\mathtt{in}\langle c\rangle.$$

The law above is true for barbed congruence, as we can prove using results of Section 4.3, but it is not for delay bisimilarity. Indeed, the action $\mathtt{in}\langle a\rangle$, leading the right hand side to $!\mathtt{in}\langle a\rangle.(\mathtt{in}\langle b\rangle \oplus \mathtt{in}\langle c\rangle) \mid \mathtt{in}\langle c\rangle$, cannot be matched by the left hand side without performing a τ action after $\mathtt{in}\langle a\rangle$. □

4.3 Ambient bisimilarity

As pointed out in the proof of Proposition 4.10 the problem of delay bisimilarity is that, by definition, weak actions $\stackrel{\alpha}{\Longrightarrow}$ do not allow τ moves after the visible action α. In this section, we give a slightly different lts $\rightarrow$ defined in terms of $\longmapsto$, which allows us to define weak actions $\stackrel{\alpha}{\Longrightarrow}$ where τ moves can follow visible actions. Actions, between processes, of the form $P \stackrel{\alpha}{\longmapsto} P'$, remain unaffected. The only modification involves higher-order actions, that is, actions of the form $P \stackrel{\alpha}{\longmapsto} K$ which will also be transformed into actions between processes; the resulting lts is in *early* style.

Definition 4.11 (Ambient Transitions). *Let m be an arbitrary name and R an arbitrary process. Then:*

- $P \xrightarrow{\mathtt{enter}\langle n,h\rangle R} K \bullet R$ *if* $P \stackrel{\mathtt{enter}\langle n,h\rangle}{\longmapsto} K$
- $P \xrightarrow{\overline{\mathtt{enter}}\langle n,h\rangle m[R]} K \bullet m[R]$ *if* $P \stackrel{\overline{\mathtt{enter}}\langle n,h\rangle}{\longmapsto} K$
- $P \xrightarrow{\mathtt{exit}\langle n,h\rangle R} K \bullet R$ *if* $P \stackrel{\mathtt{exit}\langle n,h\rangle}{\longmapsto} K$
- $P \xrightarrow{\alpha} P'$ *if* $P \stackrel{\alpha}{\longmapsto} P'$

These transitions give rise to *weak* transitions in the standard manner:

- $\stackrel{\alpha}{\Longrightarrow}$ denotes $\stackrel{\tau}{\longrightarrow}^* \stackrel{\alpha}{\longrightarrow} \stackrel{\tau}{\longrightarrow}^*$
- $\stackrel{\hat{\alpha}}{\Longrightarrow}$ denotes $\stackrel{\tau}{\longrightarrow}^*$ if $\alpha = \tau$ and $\stackrel{\alpha}{\Longrightarrow}$ otherwise.

Definition 4.12 (Ambient Bisimilarity). *A symmetric relation $\mathcal{S}$ is an* ambient bisimulation *if $P\ \mathcal{S}\ Q$ and $P \stackrel{\alpha}{\longrightarrow} P'$ implies that there exists Q' such that $Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ and $P'\ \mathcal{S}\ Q'$. P and Q are ambient bisimilar, written $P \approx Q$, if $P\ \mathcal{S}\ Q$ for some ambient bisimulation $\mathcal{S}$.*

Theorem 4.13. *Ambient bisimilarity is a congruence.*

Proof: As in the proof of Theorem 4.8 we define an equivalence relation $\mathcal{S}$ which, by construction, (i) contains the relation $\approx$, and (ii) is preserved by restriction, parallel composition, and ambient operators. Then we prove that $\mathcal{S}$ is an ambient bisimilarity up to $\equiv$, by induction on the definition of $\mathcal{S}$.

In order to emphasize the differences with the proof for delay bisimilarity we focus on one case, when $P \mid R \;\; \mathcal{S} \;\; Q \mid R$ because $P \;\; \mathcal{S} \;\; Q$. Here we now carry out an inductive analysis on the transition $P \mid R \xrightarrow{\alpha} O$. The most interesting case is when $\alpha = \tau$. We recall that $P \mid R \xrightarrow{\tau} O$ if $P \mid R \vdash\!\!\xrightarrow{\tau} O$.

Let us consider the case when $P \mid R \vdash\!\!\xrightarrow{\tau} O$ because $P \vdash\!\!\xrightarrow{\overline{\texttt{enter}}\langle n,h\rangle} K_1$ and $R \vdash\!\!\xrightarrow{\texttt{enter}\langle n,h\rangle} \boldsymbol{\nu}\tilde{r}\langle m[R_1]\rangle_n R_2$, with $O \equiv \boldsymbol{\nu}\tilde{r}\big((K_1 \bullet m[R_1]) \mid R_2\big)$ (we recall that by Lemma 3.1 only ambients can migrate). By definition, we have $P \xrightarrow{\overline{\texttt{enter}}\langle n,h\rangle m[R_1]} K_1 \bullet m[R_1] = P'$. So here we can still use the induction hypothesis, $P \;\; \mathcal{S} \;\; Q$, to find a move $Q \Rightarrow Q'$ such that $P' \;\; \mathcal{S} \;\; Q'$ where $Q \Rightarrow\!\vdash\!\!\xrightarrow{\overline{\texttt{enter}}\langle n,h\rangle} K_2$ and $K_2 \bullet m[R_1] \Rightarrow Q'$. Thus:

- $P \mid R \xrightarrow{\tau} \boldsymbol{\nu}\tilde{r}\big((K_1 \bullet m[R_1]) \mid R_2\big) \equiv O$
- $Q \mid R \Rightarrow \xrightarrow{\tau} \boldsymbol{\nu}\tilde{r}\big((K_2 \bullet m[R_1]) \mid R_2\big) \Rightarrow \boldsymbol{\nu}\tilde{r}\big(Q' \mid R_2\big) \equiv O'$.

As $P' \;\; \mathcal{S} \;\; Q'$ and $\mathcal{S}$ is preserved by parallel composition and restriction, we get $O \;\; \mathcal{S} \;\; O'$, as desired. The other cases are similar. □

With this result we can show, as for delay bisimilarity, that ambient bisimilarity $\approx$ is contained in barbed congruence $\cong$. The next step is to prove that ambient bisimilarity completely characterises barbed congruence. To this end we use two contexts $\mathrm{SPY_a}\langle n, h_1, h_2, \cdot\rangle$ and $\mathrm{SPY_b}\langle n, h_1, h_2, \cdot\rangle$ which allows us to *spy* on any process R plugged into the hole. These two contexts will be defined so that for any process R

$$\mathrm{SPY_a}\langle n, h_1, h_2, R\rangle \Downarrow_{\texttt{pop}\langle n,h_i\rangle}$$

and similarly, for any name m,

$$m[\mathrm{SPY_b}\langle n, h_1, h_2, R\rangle] \Downarrow_{\texttt{pop}\langle n,h_i\rangle}$$

for $i \in \{1, 2\}$. The ability to *spy* on R derives from the fact that one of the two barbs is lost when process R performs any action.

Definition 4.14. *We use* $\mathrm{SPY_a}\langle n, h_1, h_2, \cdot\rangle$ *and* $\mathrm{SPY_b}\langle n, h_1, h_2, \cdot\rangle$ *as shorthands for the two following contexts, respectively:*

- $\boldsymbol{\nu}(zr)\big(r[\overline{\texttt{open}}\langle r\rangle.[\cdot]] \mid (\texttt{open}\langle r\rangle \mid z[\texttt{out}\langle n, h_1\rangle]) \oplus (\texttt{open}\langle r\rangle \mid z[\texttt{out}\langle n, h_2\rangle])\big)$
- $\boldsymbol{\nu}r\big(r[\overline{\texttt{open}}\langle r\rangle.[\cdot]] \mid (\texttt{open}\langle r\rangle \mid \texttt{out}\langle n, h_1\rangle) \oplus (\texttt{open}\langle r\rangle \mid \texttt{out}\langle n, h_2\rangle)\big)$

The following result formally states the mentioned above property of the *spy cages* $\mathrm{SPY_a}\langle n, h_1, h_2, R\rangle$ and $\mathrm{SPY_b}\langle n, h_1, h_2, R\rangle$. To this end, we say that a context $C[\cdot]$ is *static* if the hole $[\cdot]$ appears only underneath the operators of restriction, parallel composition and ambient.

LEMMA 4.15. *Let $C[\cdot]$ be a static context, R a process, n a name, and h_1, h_2 fresh names. Then:*

1. *If $C[\mathrm{SPY_a}\langle n, h_1, h_2, R\rangle] \xrightarrow{\tau} P$ and $P \Downarrow_{\mathtt{pop}\langle n, h_i\rangle}$, $i \in \{1,2\}$, then there is a static context $C'[\cdot]$ such that*
   - *$P = C'[\mathrm{SPY_a}\langle n, h_1, h_2, R\rangle]$ and*
   - *$C[R] \xrightarrow{\tau} C'[R]$.*
2. *If $C[m[\![\mathrm{SPY_b}\langle n, h_1, h_2, R\rangle]\!]] \xrightarrow{\tau} P$ and $P \Downarrow_{\mathtt{pop}\langle n, h_i\rangle}$, $i \in \{1,2\}$, then there is a static context $C'[\cdot]$ such that*
   - *$P = C'[m[\![\mathrm{SPY_b}\langle n, h_1, h_2, R\rangle]\!]]$ and*
   - *$C[m[\![R]\!]] \xrightarrow{\tau} C'[m[\![R]\!]]$.*

Proof: By transition induction. □

By virtue of Theorem 4.4, when proving that $\approx$ and $\cong$ coincide it suffices to show that $\approx$ and $\cong_{\mathrm{pop}}$ coincide. To this end we need a last lemma which allows us to remove the spy cages $\mathrm{SPY_a}\langle n, h_1, h_2, \cdot\rangle$ and $\mathrm{SPY_b}\langle n, h_1, h_2, \cdot\rangle$.

LEMMA 4.16 (CUTTING LEMMA). *Let $C_1[\cdot]$ and $C_2[\cdot]$ be static contexts, P, Q and R processes, and h_1 and h_2 fresh names.*

1. *If $C_1[\mathrm{SPY_a}\langle n, h_1, h_2, R\rangle] \cong_{\mathrm{pop}} C_2[\mathrm{SPY_a}\langle n, h_1, h_2, R\rangle]$ then $C_1[R] \cong_{\mathrm{pop}} C_2[R]$.*
2. *If $C_1[m[\![\mathrm{SPY_b}\langle n, h_1, h_2, R\rangle]\!]] \cong_{\mathrm{pop}} C_2[m[\![\mathrm{SPY_b}\langle n, h_1, h_2, R\rangle]\!]]$ then $C_1[m[\![R]\!]] \cong_{\mathrm{pop}} C_2[m[\![R]\!]]$.*
3. *If $P \mid R \cong_{\mathrm{pop}} Q \mid R$, with $\mathrm{fn}(R) \cap \mathrm{fn}(P, Q) = \emptyset$, then $P \cong_{\mathrm{pop}} Q$.*

Proof: To prove Part 1 note that since $\cong_{\mathrm{pop}}$ is closed under restriction, we have that:

$$\boldsymbol{\nu}(h_1 h_2)(C_1[\mathrm{SPY_a}\langle n, h_1, h_2, R\rangle]) \cong_{\mathrm{pop}} \boldsymbol{\nu}(h_1 h_2)(C_2[\mathrm{SPY_a}\langle n, h_1, h_2, R\rangle]).$$

As h_1 and h_2 are fresh,

$$\boldsymbol{\nu}(h_1 h_2) C_i[\mathrm{SPY_a}\langle n, h_1, h_2, R\rangle] \equiv C_i[\boldsymbol{\nu}(h_1 h_2)\mathrm{SPY_a}\langle n, h_1, h_2, R\rangle]$$

for $i \in \{1,2\}$. By exhibiting an appropriate bisimulation it is easy to prove that $\boldsymbol{\nu}(h_1 h_2)\mathrm{SPY_a}\langle n, h_1, h_2, R\rangle \approx R$. As $\approx$ implies $\cong$, and, by Theorem 4.4, relations $\cong$ and $\cong_{\mathrm{pop}}$ coincide, we get $C_1[R] \cong_{\mathrm{pop}} C_2[R]$, as desired.
The proof of Part 2 is similar.
To prove Part 3 we set $\tilde{r} = \mathrm{fn}(R)$ so that $\boldsymbol{\nu}\tilde{r}R \cong_{\mathrm{pop}} \mathbf{0}$. An easy way to prove this, is by showing that $\boldsymbol{\nu}\tilde{r}R \approx \mathbf{0}$. Finally, since $\mathrm{fn}(R) \cap \mathrm{fn}(P, Q) = \emptyset$

and $\cong_{\mathrm{pop}}$ is preserved by restriction we get:

$$P \cong_{\mathrm{pop}} P \mid \boldsymbol{\nu}\tilde{r}R \cong_{\mathrm{pop}} \boldsymbol{\nu}\tilde{r}(P \mid R) \cong_{\mathrm{pop}} \boldsymbol{\nu}\tilde{r}(Q \mid R) \cong_{\mathrm{pop}} Q \mid \boldsymbol{\nu}\tilde{r}R \cong_{\mathrm{pop}} Q$$

as desired. □

THEOREM 4.17. *Ambient bisimilarity and barbed congruence coincide.*

Proof: By Theorem 4.13 and Lemma 4.1 ambient bisimilarity is contained in barbed congruence. As to the completeness part, by Theorem 4.4, it suffices to prove that the relation

$$\mathcal{S} = \{(P, Q) : P \cong_{\mathrm{pop}} Q\}$$

is an ambient bisimilarity up to $\equiv$.
Let us first consider the three possible higher-order actions $P \xrightarrow{\alpha} O$:

1. Let $P \xrightarrow{\mathtt{enter}\langle n,h\rangle R} P'$ because $P \vdash \xrightarrow{\mathtt{enter}\langle n,h\rangle} K_1 = \boldsymbol{\nu}\tilde{p}\langle P_1\rangle_n P_2$ where $P' = C_1[R]$ and $C_1[\cdot] = \boldsymbol{\nu}\tilde{p}(n[[\cdot] \mid P_1] \mid P_2)$. We want to conclude that there is a matching move $Q \xRightarrow{\mathtt{enter}\langle n,h\rangle R} Q'$ with $P' \ \mathcal{S} \ Q'$. We define:

$$C_{\alpha R}[\cdot] \stackrel{\mathrm{def}}{=} [\cdot] \mid n[\overline{\mathtt{in}}\langle n,h\rangle.(\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle \oplus a[\mathtt{out}\langle n,h_3\rangle])]$$

with a, h_i fresh. As $P \cong_{\mathrm{pop}} Q$ it follows that $C_{\alpha R}[P] \cong_{\mathrm{pop}} C_{\alpha R}[Q]$. So, if

$$C_{\alpha R}[P] \xrightarrow{\tau} \xrightarrow{\tau} C_1[\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle]$$

then there is a process Z such that

$$C_{\alpha R}[Q] \Rightarrow Z \text{ and } C_1[\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle] \cong_{\mathrm{pop}} Z.$$

As a consequence, $Z{\Downarrow}_{\mathtt{pop}\langle n,h_1\rangle}$, $Z{\Downarrow}_{\mathtt{pop}\langle n,h_2\rangle}$, and $Z{\not\Downarrow}_{\mathtt{pop}\langle n,h_3\rangle}$. This implies that in the reductions sequence $C_{\alpha R}[Q] \Rightarrow Z$ the prefix $\overline{\mathtt{in}}\langle n,h\rangle$ is consumed. More precisely, by Lemma 4.15(1) there exist static contexts $C'[\cdot], C''[\cdot]$ and $C_2[\cdot]$ such that:

$$\begin{array}{rll} C_{\alpha R}[Q] = & & Q \mid n[\overline{\mathtt{in}}\langle n,h\rangle.(\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle \oplus a[\mathtt{out}\langle n,h_3\rangle])] \\ & \Longrightarrow \xrightarrow{\tau} & C'[\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle \oplus a[\mathtt{out}\langle n,h_3\rangle]] \\ & \Longrightarrow \xrightarrow{\tau} & C''[\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle] \\ & \Longrightarrow & C_2[\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle] \\ & = & Z \end{array}$$

By Lemma 4.16(1), we have $C_1[R] \cong_{\mathrm{pop}} C_2[R]$, that is $P' \cong_{\mathrm{pop}} C_2[R]$. It remains to show that $Q \xRightarrow{\mathtt{enter}\langle n,h\rangle R} C_2[R]$, that is the required Q' is $C_2[R]$.

Examining the above reductions sequence from $C_{\alpha R}[Q]$ we know that $Q \Rightarrow \xrightarrow{\mathtt{enter}\langle n,h\rangle R} C'[R]$. We also know that $C'[(\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle)] \stackrel{\tau}{\Longrightarrow} C_2[(\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle)]$. Repeated application of Lemma 4.15(1) gives $C'[R] \Rightarrow C_2[R]$, and therefore we have the required corresponding action $Q \xRightarrow{\mathtt{enter}\langle n,h\rangle R} C_2[R]$.

2. Let $P \xrightarrow{\mathtt{exit}\langle n,h\rangle R} P'$ because $P \vdash \xrightarrow{\mathtt{exit}\langle n,h\rangle} K_1 = \boldsymbol{\nu}\tilde{p}\langle P_1\rangle_n P_2$ where $P' = C_1[R]$ and $C_1[\cdot] = \boldsymbol{\nu}\tilde{p}(n[[\cdot] \mid P_1] \mid P_2)$. Again we want to conclude that there is a a matching move such that $Q \xRightarrow{\mathtt{exit}\langle n,h\rangle R} Q'$ with $P' \;\mathcal{S}\; Q'$. The proof strategy is the same as in the first case except that here we use the context

$$C_{\alpha R}[\cdot] \stackrel{\text{def}}{=} n[[\cdot] \mid \mathrm{SPY_a}\langle n,h_1,h_2,R\rangle] \mid \overline{\mathtt{out}}\langle n,h\rangle.(a[b[\mathtt{out}\langle a,h_3\rangle]] \oplus a[b[\mathtt{out}\langle a,h_4\rangle]])$$

with a, b, h_i fresh. Again we have $C_{\alpha R}[P] \cong_{\mathrm{pop}} C_{\alpha R}[Q]$. So, if

$$C_{\alpha R}[P] \xrightarrow{\tau} \xrightarrow{\tau} C_1[\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle] \mid a[b[\mathtt{out}\langle a,h_3\rangle]]$$

then there is a process Z such that

$$C_{\alpha R}[Q] \Rightarrow Z \text{ and } C_1[\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle] \mid a[b[\mathtt{out}\langle a,h_3\rangle]] \cong_{\mathrm{pop}} Z.$$

As a consequence, $Z{\Downarrow_{\mathtt{pop}\langle n,h_1\rangle}}$, $Z{\Downarrow_{\mathtt{pop}\langle n,h_2\rangle}}$, $Z{\Downarrow_{\mathtt{pop}\langle a,h_3\rangle}}$, and $Z{\not\Downarrow_{\mathtt{pop}\langle a,h_4\rangle}}$. This implies that in the reductions sequence $C_{\alpha R}[Q] \Rightarrow Z$ the prefix $\overline{\mathtt{out}}\langle n,h\rangle$ is consumed. More precisely, by Lemma 4.15(1) there exist static contexts $C'[\cdot], C''[\cdot]$ and $C_2[\cdot]$ such that:

$$\begin{array}{lll}
C_{\alpha R}[Q] = & & n[Q \mid \mathrm{SPY_a}\langle n,h_1,h_2,R\rangle] \mid \\
 & & \overline{\mathtt{out}}\langle n,h\rangle.(a[b[\mathtt{out}\langle a,h_3\rangle]] \oplus a[b[\mathtt{out}\langle a,h_4\rangle]]) \\
 & \Longrightarrow \xrightarrow{\tau} & C'[\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle] \mid (a[b[\mathtt{out}\langle a,h_3\rangle]] \oplus a[b[\mathtt{out}\langle a,h_4\rangle]]) \\
 & \Longrightarrow \xrightarrow{\tau} & C''[\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle] \mid a[b[\mathtt{out}\langle a,h_3\rangle]] \\
 & \Longrightarrow & C_2[\mathrm{SPY_a}\langle n,h_1,h_2,R\rangle] \mid a[b[\mathtt{out}\langle a,h_3\rangle]] \\
 & = & Z
\end{array}$$

By Lemmas 4.16(1) and 4.16(3), we have $C_1[R] \cong_{\mathrm{pop}} C_2[R]$.

As in the first case we can now show that the required Q' is $C_2[R]$; analysing the above reduction and applying Lemma 4.15(1) we obtain $Q \xRightarrow{\mathtt{exit}\langle n,h\rangle R} Q'$, as required.

3. Let $P \xrightarrow{\overline{\mathtt{enter}}\langle n,h\rangle m[R]} P'$, that is $P \vdash \xrightarrow{\overline{\mathtt{enter}}\langle n,h\rangle} K_1 = \boldsymbol{\nu}\tilde{p}\langle P_1\rangle_n P_2$ where $P' = C_1[m[R]]$ and $C_1[\cdot] = \boldsymbol{\nu}\tilde{p}(n[[\cdot] \mid P_1] \mid P_2)$. Here we need to find some Q' such that $Q \xRightarrow{\overline{\mathtt{enter}}\langle n,h\rangle m[R]} Q'$ and $P' \;\mathcal{S}\; Q'$. This

time we use the context

$$C_{\alpha m[R]}[\cdot] \stackrel{\text{def}}{=} [\cdot] \mid m[\mathtt{in}\langle n,h\rangle.(\mathrm{SPY_b}\langle n,h_1,h_2,R\rangle \oplus \mathtt{out}\langle n,h_3\rangle)]$$

with h_i fresh. Arguing as usual from $C_{\alpha m[R]}[P] \cong_{\text{pop}} C_{\alpha m[R]}[Q]$, we know that since

$$C_{\alpha m[R]}[P] \xrightarrow{\tau} \xrightarrow{\tau} C_1[m[\mathrm{SPY_b}\langle n,h_1,h_2,R\rangle]],$$

there is a process Z such that

$$C_{\alpha m[R]}[Q] \Rightarrow Z \text{ and } C_1[m[\mathrm{SPY_b}\langle n,h_1,h_2,R\rangle]] \cong_{\text{pop}} Z.$$

As a consequence, $Z{\Downarrow_{\mathtt{pop}\langle n,h_1\rangle}}$, $Z{\Downarrow_{\mathtt{pop}\langle n,h_2\rangle}}$, and $Z{\not\Downarrow_{\mathtt{pop}\langle n,h_3\rangle}}$. This implies that in the reductions sequence $C_{\alpha m[R]}[Q] \Rightarrow Z$ the prefix $\mathtt{in}\langle n,h\rangle$ is consumed. More precisely, this time by Lemma 4.15(2), there exist static contexts $C'[\cdot], C''[\cdot]$ and $C_2[\cdot]$ such that:

$$\begin{array}{rll} C_{\alpha m[R]}[Q] = & & Q \mid m[\mathtt{in}\langle n,h\rangle.(\mathrm{SPY_b}\langle n,h_1,h_2,R\rangle \oplus \mathtt{out}\langle n,h_3\rangle)] \\ & \Longrightarrow \xrightarrow{\tau} & C'[m[\mathrm{SPY_b}\langle n,h_1,h_2,R\rangle \oplus \mathtt{out}\langle n,h_3\rangle]] \\ & \Longrightarrow \xrightarrow{\tau} & C''[m[\mathrm{SPY_b}\langle n,h_1,h_2,R\rangle]] \\ & \Longrightarrow & C_2[m[\mathrm{SPY_b}\langle n,h_1,h_2,R\rangle]] \\ & = & Z \end{array}$$

By Lemma 4.16(2) we have $C_1[m[R]] \cong_{\text{pop}} C_2[m[R]]$.

We now have the required Q', namely $C_2[m[R]]$. An analysis of the above reductions gives $Q \Rightarrow \xrightarrow{\overline{\mathtt{enter}}\langle n,h\rangle m[R]} C'[m[R]]$, and from Lemma 4.15(2) we know that $C'[m[R]] \Rightarrow Q'$. We therefore have the required move $Q \xRightarrow{\overline{\mathtt{enter}}\langle n,h\rangle m[R]} Q'$.

The remaining cases concern the simpler actions of the form $P \xrightarrow{\alpha} P'$ where P' is a process; there are eight cases in all. Here it will be useful to write $h \oplus h'$ as an abbreviation for $f[\boldsymbol{\nu} z(z[\mathtt{out}\langle f,h\rangle])] \oplus f[\boldsymbol{\nu} z(z[\mathtt{out}\langle f,h'\rangle])]$ where f is always assumed to be fresh.

- Let $P \xrightarrow{\mathtt{in}\langle n,h\rangle} P'$. We want to conclude that there is Q' such that $Q \xRightarrow{\mathtt{in}\langle n,h\rangle} Q'$ and $P' \ \mathcal{S} \ Q'$. We define:

$$\begin{array}{rl} C_\alpha[\cdot] \stackrel{\text{def}}{=} & a[[\cdot] \mid \mathtt{out}\langle n,h_1\rangle.\overline{\mathtt{open}}\langle a\rangle] \mid n[\overline{\mathtt{in}}\langle n,h\rangle] \mid \\ & \overline{\mathtt{out}}\langle n,h_1\rangle.\mathtt{open}\langle a\rangle.(h_2 \oplus h_3) \end{array}$$

with a, h_i fresh. From $P \cong_{\text{pop}} Q$ we know that $C_\alpha[P] \cong_{\text{pop}} C_\alpha[Q]$. So, if

$$C_\alpha[P] \xrightarrow{\tau} \xrightarrow{\tau} \xrightarrow{\tau} \xrightarrow{\tau} P' \mid n[\ ] \mid h_2 \cong_{\text{pop}} P' \mid h_2$$

then there is a process Z such that $P' \mid h_2 \cong_{\text{pop}} Z$. As a consequence,

$Z \Downarrow_{\mathtt{pop}\langle f,h_2\rangle}$ and $Z \not\Downarrow_{\mathtt{pop}\langle f,h_3\rangle}$. This implies that in the reductions sequence $C_\alpha[Q] \Rightarrow Z$ the whole context $C_\alpha[\cdot]$ is consumed (up to $\cong_{\mathrm{pop}}$) except for h_2. More precisely, noticing that $n[\,] \cong_{\mathrm{pop}} \mathbf{0}$ (see Section 6), there exist Q_1, Q_2, Q_3, Q' such that:

$$\begin{array}{lll} C_\alpha[Q] = & & a[Q \mid \mathtt{out}\langle n,h_1\rangle.\overline{\mathtt{open}}\langle a\rangle] \mid n[\overline{\mathtt{in}}\langle n,h\rangle] \mid \\ & & \overline{\mathtt{out}}\langle n,h_1\rangle.\mathtt{open}\langle a\rangle.(h_2 \oplus h_3) \\ & \Longrightarrow \stackrel{\tau}{\longrightarrow} & n[a[Q_1 \mid \mathtt{out}\langle n,h_1\rangle.\overline{\mathtt{open}}\langle a\rangle]] \mid \\ & & \overline{\mathtt{out}}\langle n,h_1\rangle.\mathtt{open}\langle a\rangle.(h_2 \oplus h_3) \\ & \Longrightarrow \stackrel{\tau}{\longrightarrow} & a[Q_2 \mid \overline{\mathtt{open}}\langle a\rangle] \mid n[\,] \mid \mathtt{open}\langle a\rangle.(h_2 \oplus h_3) \\ & \Longrightarrow \stackrel{\tau}{\longrightarrow} & Q_3 \mid n[\,] \mid (h_2 \oplus h_3) \\ & \Longrightarrow & Q' \mid n[\,] \mid h_2 \\ & = & Z \\ & \cong_{\mathrm{pop}} & Q' \mid h_2 \end{array}$$

where $Q \Rightarrow \stackrel{\mathtt{in}\langle n,h\rangle}{\longrightarrow} Q_1 \Rightarrow Q_2 \Rightarrow Q_3 \Rightarrow Q'$. By Lemma 4.16(3) we can conclude that $P' \cong_{\mathrm{pop}} Q'$, as desired.

- Let $P \stackrel{\overline{\mathtt{in}}\langle n,h\rangle}{\longrightarrow} P'$. Again we need to find some Q' such that $Q \stackrel{\overline{\mathtt{in}}\langle n,h\rangle}{\Longrightarrow} Q'$ and $P' \;\mathcal{S}\; Q'$. The argument is the same as in the previous case, this time using the context

$$\begin{array}{rcl} C_\alpha[\cdot] & \stackrel{\mathrm{def}}{=} & n[[\cdot] \mid \overline{\mathtt{open}}\langle n,h_1\rangle.(h_2 \oplus h_3)] \mid \\ & & a[\mathtt{in}\langle n,h\rangle.\mathtt{out}\langle n,h_4\rangle] \mid \overline{\mathtt{out}}\langle n,h_4\rangle.\mathtt{open}\langle n,h_1\rangle \end{array}$$

with a, h_i fresh. From the move $C_\alpha[P] \cong_{\mathrm{pop}} C_\alpha[Q]$. So, if

$$C_\alpha[P] \stackrel{\tau}{\longrightarrow} \stackrel{\tau}{\longrightarrow} \stackrel{\tau}{\longrightarrow} \stackrel{\tau}{\longrightarrow} P' \mid a[\,] \mid h_2 \cong_{\mathrm{pop}} P' \mid h_2$$

we know that there is a process Z such that $P' \mid h_2 \cong_{\mathrm{pop}} Z$. As a consequence, $Z \Downarrow_{\mathtt{pop}\langle f,h_2\rangle}$ whereas $Z \not\Downarrow_{\mathtt{pop}\langle f,h_3\rangle}$. This implies that in the reduction sequence $C_\alpha[Q] \Rightarrow Z$ the whole context $C_\alpha[\cdot]$ is consumed (up to $\cong_{\mathrm{pop}}$) except for h_2. More precisely, noticing that $a[\,] \cong_{\mathrm{pop}} \mathbf{0}$,

there exist Q_1, Q_2, and Q_3 such that:

$$\begin{array}{lll} C_\alpha[Q] = & & n[Q \mid \overline{\mathtt{open}}\langle n, h_1\rangle.(h_2 \oplus h_3)] \mid a[\mathtt{in}\langle n, h\rangle.\mathtt{out}\langle n, h_4\rangle] \mid \\ & & \overline{\mathtt{out}}\langle n, h_4\rangle.\mathtt{open}\langle n, h_1\rangle \\ & \Longrightarrow \xrightarrow{\tau} & n[Q_1 \mid \overline{\mathtt{open}}\langle n, h_1\rangle.(h_2 \oplus h_3) \mid a[\mathtt{out}\langle n, h_4\rangle]] \mid \\ & & \overline{\mathtt{out}}\langle n, h_4\rangle.\mathtt{open}\langle n, h_1\rangle \\ & \Longrightarrow \xrightarrow{\tau} & n[Q_2 \mid \overline{\mathtt{open}}\langle n, h_1\rangle.(h_2 \oplus h_3)] \mid a[\ ] \mid \mathtt{open}\langle n, h_1\rangle \\ & \Longrightarrow \xrightarrow{\tau} & Q_3 \mid (h_2 \oplus h_3) \mid a[\ ] \\ & \Longrightarrow \xrightarrow{\tau} & Q' \mid h_2 \mid a[\ ] \\ & = & Z \\ & \cong_{\mathrm{pop}} & Q' \mid h_2 \end{array}$$

where $Q \Rightarrow \xrightarrow{\overline{\mathtt{in}}\langle n,h\rangle} Q_1 \Rightarrow Q_2 \Rightarrow Q_3 \Rightarrow Q'$. By Lemma 4.16(3) we can conclude that $P' \cong_{\mathrm{pop}} Q'$, as desired.

- The six remaining cases are similar, except that we need an appropriate context. These are detailed as follows:
  - for $\alpha = \mathtt{pop}\langle n, h\rangle$ use
    $$C_\alpha[\cdot] \stackrel{\mathrm{def}}{=} [\cdot] \mid \overline{\mathtt{out}}\langle n, h\rangle.(h_1 \oplus h_2)$$
    with h_1 and h_2 fresh.
  - for $\alpha = \overline{\mathtt{out}}\langle n, h\rangle$ use
    $$C_\alpha[\cdot] \stackrel{\mathrm{def}}{=} [\cdot] \mid n[a[\mathtt{out}\langle n, h\rangle.\overline{\mathtt{open}}\langle a\rangle]] \mid \mathtt{open}\langle a\rangle.(h_1 \oplus h_2)$$
    with a, h_1 and h_2 fresh.
  - for $\alpha = \mathtt{out}\langle n, h\rangle$ use
    $$C_\alpha[\cdot] \stackrel{\mathrm{def}}{=} n[a[[\cdot] \mid \overline{\mathtt{open}}\langle a\rangle]] \mid \overline{\mathtt{out}}\langle n, h\rangle.\mathtt{open}\langle a\rangle.(h_1 \oplus h_2)$$
    with a, h_1 and h_2 fresh.
  - for $\alpha = \mathtt{open}\langle n, h\rangle$ use
    $$C_\alpha[\cdot] \stackrel{\mathrm{def}}{=} [\cdot] \mid n[\overline{\mathtt{open}}\langle n, h\rangle.(h_1 \oplus h_2)]$$
    with h_1 and h_2 fresh.
  - for $\alpha = \overline{\mathtt{open}}\langle n, h\rangle$ use
    $$C_\alpha[\cdot] \stackrel{\mathrm{def}}{=} n[[\cdot]] \mid \mathtt{open}\langle n, h\rangle.(h_1 \oplus h_2)$$
    with h_1 and h_2 fresh.
  - for $\alpha = \mathtt{free}\langle n, h\rangle$ use
    $$C_\alpha[\cdot] \stackrel{\mathrm{def}}{=} [\cdot] \mid \mathtt{open}\langle n, h\rangle.(h_1 \oplus h_2)$$

with h_1 and h_2 fresh.

□

We believe that the distinguishing contexts in the proof above can be defined without the use of passwords, except when α is a $\overline{\texttt{enter}}$ action. In this case however the use of fresh passwords is essential. In order to test that a process can allow entry to an ambient we can send it an ambient which contains a fresh password. Probing for this fresh password ensures that the ambient we have sent has indeed been accepted. Without fresh passwords there would be no distinguishing feature of the ambient sent which could be used in the probe.

Note also that our rules for $\overline{\texttt{out}}$, different from those in [9], have played a crucial role in the distinguishing contexts for both $\overline{\texttt{enter}}$ and $\overline{\texttt{in}}$. The alternative semantics for $\overline{\texttt{out}}\langle n\rangle$ given in [9] uses an auxiliary action $?n$ but it is difficult to conceive of a distinguishing context for this action.

5 Adding Communication

Both Mobile Ambients, [6], and Safe Ambients, [9], allow local communication inside ambients. The basic idea is to have an *output process* such as $\langle E\rangle$, which outputs the message E, and an input process such as $(x).Q$ which on receiving a message binds it to x in Q which then executes. The basic reduction rule therefore takes the form

$$(x).Q \mid \langle E\rangle \ \rightarrow Q\{^{E}/_{x}\} \mid P$$

In this section we show that our results can be extended to such a message-passing language.

The syntax of our extended language is given in Table 7. The prefixing operator $C.P$ of Section 2 is generalised to $G.P$, where G is a syntactic category of *guards*. This may take the form

- $E.P$, a direct generalisation of $C.P$. Here E is any *path*, or sequence, of capabilities. These paths will be the messages allowed in our systems.
- $\langle E\rangle.P$, representing the synchronous output of the message E; the process P can not be executed until the message E has been consumed. As discussed in [21, 2] this is not unrealistic because communication is always local.
- $(x).Q$, representing input of a message to be bound to x in Q.

We now have variables in our language, with the construct $(x).Q$ a binding construct for x. This gives rise in the standard manner to the notions of free and bound variables, $\mathrm{fv}(\cdot)$ and $\mathrm{bv}(\cdot)$, α-equivalence and

TABLE 7 The Message-passing Calculus SAP

Names: $n, h, \ldots \in \mathbf{N}$

Variables: $x, y, \ldots \in \mathbf{X}$

Capabilities:

C	::=	$\mathtt{in}\langle n, h\rangle$	may enter into n
	|	$\mathtt{out}\langle n, h\rangle$	may exit out of n
	|	$\mathtt{open}\langle n, h\rangle$	may open n
	|	$\overline{\mathtt{in}}\langle n, h\rangle$	allow enter
	|	$\overline{\mathtt{out}}\langle n, h\rangle$	allow exit
	|	$\overline{\mathtt{open}}\langle n, h\rangle$	allow open

Expressions:

E, F	::=	x	variable
	|	C	capability
	|	$E.F$	path
	|	ϵ	empty path

Guards:

G	::=	E	expression
	|	(x)	input
	|	$\langle E\rangle$	output

Processes:

P	::=	$\mathbf{0}$	nil process
	|	$P_1 \mid P_2$	parallel composition
	|	$\boldsymbol{\nu} n P$	restriction
	|	$G.P$	prefixing
	|	$n[P]$	ambient
	|	$!G.P$	replication

Concretions:

P	::=	$\boldsymbol{\nu}\tilde{m}\langle P\rangle_n Q$	movement concretion
	|	$\boldsymbol{\nu}\tilde{p}\langle E\rangle P$	buffer concretion

substitutions in which free occurrences of variables are not captured; we avoid spelling out the details. A process is now any closed term, that is any P in which $\mathrm{fv}(P) = \emptyset$.

The operational semantics is defined by introducing two new labels for input and output transitions and a new form of concretion:

- $P \stackrel{(E)}{\longmapsto} Q$ means that the process P can receive the message E and continue as Q
- $P \stackrel{\langle - \rangle}{\longmapsto} \boldsymbol{\nu}\tilde{p}\langle E \rangle Q$ means that P can output the message E which shares the bound names $\tilde{p}$ with the residual Q.

Other possibilities for formalising message transmission are discussed at the end of this section.

In Table 8 we give the defining rules for the operational semantics of these constructs, which should be added to those of Tables 5 and 6 to obtain the lts $P \stackrel{\mu}{\longmapsto} O$ for the processes, that is closed terms, of our extended language. The rules are straightforward and require no comment. However note that in the structural rules of Table 6 we are now assuming that parallel composition and restriction distribute over the new forms of concretions $\boldsymbol{\nu}\tilde{p}\langle E \rangle P$ in the same manner as $\boldsymbol{\nu}\tilde{p}\langle P \rangle_n Q$.

In order to obtain a reasonable semantic equivalence we must now transform these transitions into ones which do not involve concretions. The only problem is the output rule, which delivers a new form of concretion. First we define the application of these concretions to terms; this is then used, as in Section 4, to transform a transition $P \stackrel{\langle - \rangle}{\longmapsto} \boldsymbol{\nu}\tilde{p}\langle E \rangle Q$ into a transition $P \stackrel{L}{\longrightarrow} P'$ for some process P' and some label L.

Let R be any term such that $\mathrm{fv}(R) = \{x\}$; intuitively here x represents the placeholder for the message E which will be received via an output action $P \stackrel{\langle - \rangle}{\longmapsto} \boldsymbol{\nu}\tilde{p}\langle E \rangle Q$. Then, we define

$$\boldsymbol{\nu}\tilde{p}\langle E \rangle P \bullet R \stackrel{\mathrm{def}}{=} \boldsymbol{\nu}\tilde{p}(P \mid R\{^{E}/_{x}\})$$

where the bound variables $\tilde{p}$ are chosen so that $\mathrm{fn}(R) \cap \tilde{p} = \emptyset$.

We can now define the extra ambient transition, $\stackrel{\langle - \rangle R}{\longrightarrow}$, for any such R to add to those in Definition 4.11. Let

$$P \stackrel{\langle - \rangle R}{\longrightarrow} K \bullet R \text{ if } P \stackrel{\langle - \rangle}{\longmapsto} K.$$

Using Definition 4.12 with this extended set of ambient transitions we obtain ambient bisimilarity, $\approx$, between processes in our extended message-passing language.

We now outline how to extend Theorem 4.17 to this setting, relating bisimilarity to a contextual equivalence. First let us be quite precise as to how Definition 2.2 extends to our language with open and closed terms.

TABLE 8 Labelled Transition System - Communication

$$\text{(Output)}\ \frac{-}{\langle E\rangle.P \stackrel{\langle - \rangle}{\longmapsto} \langle E\rangle P} \qquad \text{(Input)}\ \frac{-}{(x).P \stackrel{(E)}{\longmapsto} P\{^{E}/_{x}\}}$$

$$\text{(Path)}\ \frac{E.(F.P) \stackrel{\alpha}{\longmapsto} Q}{(E.F).P \stackrel{\alpha}{\longmapsto} Q} \qquad (\tau\ \text{Eps})\ \frac{-}{\epsilon.P \stackrel{\tau}{\longmapsto} P}$$

$$(\tau\ \text{Comm})\ \frac{P \stackrel{\langle - \rangle}{\longmapsto} \boldsymbol{\nu}\tilde{p}\langle E\rangle P' \quad Q \stackrel{(E)}{\longmapsto} Q' \quad \text{fn}(Q') \cap \{\tilde{p}\} = \emptyset}{P \mid Q \stackrel{\tau}{\longmapsto} \boldsymbol{\nu}\tilde{p}(P' \mid Q')}$$

DEFINITION 5.1 (BARBED CONGRUENCE). *Barbed congruence, $\cong$ is the largest equivalence relation over arbitrary terms which*

- *is preserved by contexts*
- *when restricted to processes is reduction closed*
- *when restricted to processes is barb preserving.*

Defined in this manner there is an immediate mismatch between $\cong$ and the bisimulation equivalence $\approx$; the former is defined for arbitrary terms while the latter only applies to processes. However we can rectify this by generalising $\approx$ to arbitrary terms in the standard manner. For any two terms P, Q we write $P \approx Q$ if for all substitutions σ, mappings from variables to names, we have $P\sigma \approx Q\sigma$.

THEOREM 5.2. *The relation $\approx$ is a congruence for the message passing language.*

Proof: A straightforward extension of Theorem 4.13 considering each operator in turn. For example to show that it is preserved by input prefixing it is sufficient to show that

$$P \approx Q \ \text{ implies } \ (x).P \approx (x).Q$$

for all terms such that $\text{fv}(P) \cup \text{fv}(Q) \subseteq \{x\}$. However the hypothesis says that $P\{^{E}/_{x}\} \approx Q\{^{E}/_{x}\}$ for arbitrary messages E which is all that is required to prove the conclusion; the only possible moves from $(x).P$ are of the form $(x).P \stackrel{(E)}{\longrightarrow} P\{^{E}/_{x}\}$.
For all other operators it is sufficient to consider only closed terms and so the reasoning is very similar to that in Theorem 4.13. The main novelty consists in using the communication rule (τ Comm) to prove that $\approx$ is preserved by parallel composition. □

This, together with the straightforward extension of Lemma 4.1 to the message-passing calculus, immediately establishes that $\approx$ is contained in $\cong$. In fact the converse is also true.

THEOREM 5.3. *Relations $\cong$ and $\approx$ coincide over arbitrary terms in the message-passing language.*

Proof: For *processes* the proof that $\cong$ is contained in $\approx$ follows from a straightforward extension of Theorem 4.4 to the message-passing calculus. It suffices to prove that the relation

$$\mathcal{S} = \{(P,Q) : P \cong_{\rm pop} Q, \;\; P,Q \;\text{ processes}\}$$

is a bisimilarity up to $\equiv$. The main difference with respect to the proof of Theorem 4.17 is that we have to consider the cases for input and output actions.

- Let $P \xrightarrow{(M)} P'$; we want to conclude that there is Q' such that $Q \stackrel{(M)}{\Longrightarrow} Q'$ and $P' \;\; \mathcal{S} \;\; Q'$. As a distinguishing context take:

$$C_\alpha[\cdot] \stackrel{\rm def}{=} [\cdot] \mid \langle M\rangle.(h_1 \oplus h_2)$$

  with h_1 and h_2 fresh.

- Let $P \xrightarrow{\langle - \rangle R} P'$; we want to conclude that there is Q' such that $Q \stackrel{\langle - \rangle R}{\Longrightarrow} Q'$ and $P' \;\; \mathcal{S} \;\; Q'$. As a distinguishing context take:

$$C_{\alpha R}[\cdot] \stackrel{\rm def}{=} [\cdot] \mid (x).({\rm SPY_a}\langle a, h_1, h_2, R\rangle \oplus a[b[\mathtt{out}\langle a, h_3\rangle]]))$$

  with a, b, h_i fresh.

So we can conclude that for processes $P \cong Q$ implies $P \approx Q$.
Now consider two arbitrary terms P, Q such that $P \cong Q$. We need to show that $P\sigma \approx Q\sigma$ for any substitution σ. Let $x_1, x_2, \ldots, x_n$ be all the variables free in both P and Q. From $P \cong Q$ we know that $(x_1).(x_2).\ldots.(x_n).P \cong (x_1).(x_2).\ldots.(x_n).Q$ and since these are closed terms we can conclude that $(x_1).(x_2).\ldots.(x_n).P \approx (x_1).(x_2).\ldots.(x_n).Q$. But now examining the behaviour of these processes with respect to the input actions $(\sigma(x_1)), (\sigma(x_2)), \ldots$ we can conclude that $P\sigma \approx Q\sigma$ $\Box$

We end this section with two comments on our version of message-passing. First notice that we have a more restricted form of message than in [6, 9]. In particular we do not allow ambient names to be transmitted. This has been a deliberate choice as, apriori, when the name is transmitted the recipient gets considerable control over that ambient. Moreover, much of the power of name transmission can still be captured in our language.

For example the transmission of an ambient name into a static context, as in

$$\langle n\rangle.P \mid (x).\boldsymbol{\nu}\tilde{m}(x[R] \mid Q) \xrightarrow{\tau} P \quad | \quad \boldsymbol{\nu}\tilde{m}(n[R] \mid Q)$$

with $x \notin \mathrm{fv}(Q)$, can be easily emulated in our language, using passwords, by writing

$$\begin{array}{c}(y).(z).(P \mid n[y.z]) \mid \boldsymbol{\nu}(ha)(\langle\overline{\mathtt{in}}\langle n,h\rangle\rangle.\langle\mathtt{open}\langle a\rangle\rangle.\boldsymbol{\nu}\tilde{m}(a[\mathtt{in}\langle n,h\rangle.\overline{\mathtt{open}}\langle a\rangle.R] \mid Q)) \\ \xrightarrow{\tau}\xrightarrow{\tau}\cong \\ P \quad | \quad \boldsymbol{\nu}\tilde{m}(n[R] \mid Q)\end{array}$$

where $a, h \notin \mathrm{fn}(Q, R)$.

The second comment regards the formalisation of output actions using $\boldsymbol{\nu}\tilde{n}\langle M\rangle P$. This in itself is complicated but it also leads to considerably more complicated output actions in the derived lts of ambient transitions; they are higher-order, as they are parameterised on the term R.

The reader might think of avoiding these by using a simpler rule for output actions such as

$$\text{(Wrong Output)} \quad \frac{-}{\langle E\rangle.P \xrightarrow{\langle E\rangle} P}$$

Unfortunately with the rule above we are obliged to introduce bound output actions of the form $\xrightarrow{\boldsymbol{\nu}n\langle\mathtt{in}\langle n,h\rangle\rangle}$. Unfortunately, these actions are not observable. That is, there does not exist a context which is capable of recognising whether or not a process can perform the action $\xrightarrow{\boldsymbol{\nu}n\langle\mathtt{in}\langle n,h\rangle\rangle}$. Intuitively, this is because the name n is private and no context may use it as an ambient name to recognise the action $\boldsymbol{\nu}n\langle\mathtt{in}\langle n,h\rangle\rangle$. This implies that the resulting ambient bisimilarity would not coincide with barbed congruence.

6 Examples

In this section we briefly outline how our results could form the basis for reasoning techniques for ambients.

First of all our language is expressive. By simply using the names of ambients as passwords we can consider the language of Safe ambients [9] as a sub-language, although the semantics of the `out` is slightly different. Thus the various examples programmed in that paper could now be be analysised using our bisimulations. We also give two examples, similar to those given in [6, 9], which show that the existence of passwords can be of help when designing ambients.

ROUTABLE PACKETS: In [6], Cardelli and Gordon present a protocol to route a packet to various destinations. The content P and the destination E are contained in an ambient *route*. The act of sending P to destination E is realized by the following steps. Ambient *route* enters inside the packet and is opened. This liberates a message $\langle E \rangle$, which is then consumed so that the path E can be executed. At the end, the packet, which contains P, has reached the destination. Here is the program in Mobile Ambients:

$$PKT \stackrel{\text{def}}{=} pkt[!(x).x \mid !\mathtt{open}\langle route \rangle] \qquad \text{(the packet)}$$

$$\langle P, E \rangle \stackrel{\text{def}}{=} route[\mathtt{in}\langle pkt \rangle.\langle E \rangle \mid P] \qquad (P \text{ is routed to destination } E)$$

As already pointed out in [9], the protocol above works only under severe constraint on process P and on the environment. Possible dangers are:

1. process P may interfere with the path to follow;
2. two routers might enter *pkt* and interfere with the path to follow;
3. *pkt* and *route* might be opened by the environment.

These three problems are addressed in [9] by providing a new protocol along the lines of the taxi protocol in [4]. Below we adapt Levi and Sangiorgi's protocol making use of passwords. We replace $\langle P, E \rangle$ with $\langle P, E, k \rangle$ where k represents the password that must be used by the target ambient to open, and therefore access, the desired packet. Passwords allows the target ambient to distinguish between different packets addressed to it. For the sake of simplicity we rename ambients *pkt* and *route* with p and r, respectively. Moreover, as in [6], to avoid interferences from P on the path to follow, we enclose P in an ambient d.

$$PKT \stackrel{\text{def}}{=} !p[\,\overline{\mathtt{in}}\langle p \rangle.\mathtt{open}\langle r \rangle.(x).x\,]$$

$$\langle P, E, k \rangle \stackrel{\text{def}}{=} (\boldsymbol{\nu} d)r[\,\mathtt{in}\langle p \rangle.\overline{\mathtt{open}}\langle r \rangle.\langle E.\mathtt{open}\langle d \rangle \rangle.d[\overline{\mathtt{open}}\langle d \rangle.\overline{\mathtt{open}}\langle p, k \rangle.P]\,]$$

Notice that in our protocol, unlike [6, 9], the ambient p is replicated to increase the parallelism. Now, an ambient p represents a one-time "envelope" to deliver a package P at destination E. The "envelope" p is opened by the recipient by means of the password k. Notice that this example uses full replication but it can be easily rewritten in terms of replicated prefixing.

CROSSING A FIREWALL A protocol is discussed in [6] for controlling accesses through a firewall. Again our version is inspired by that in [9] but now passwords are used. Ambient f represents the firewall and h_f is

the password to cross it; ambient a represents a trusted agent inside which is a process Q that is supposed to cross the firewall. h_a is the password to access a. The firewall sends into the agent a pilot ambient k with the ability $\mathtt{in}\langle f, h_f\rangle$ to enter the firewall. The agent acquires the capability by opening k and then enters f. The process Q carried by the agent is finally liberated inside the firewall by the opening of ambient a. Here is the protocol:

$$FW \stackrel{\text{def}}{=} \boldsymbol{\nu} h_f \big(f[\ \overline{\mathtt{in}}\langle f, h_f\rangle.\mathtt{open}\langle a\rangle.P \mid k[\mathtt{out}\langle f, h_f\rangle.\mathtt{in}\langle a, h_a\rangle.\overline{\mathtt{open}}\langle k\rangle.\langle \mathtt{in}\langle f, h_f\rangle\rangle]\ ] \mid \overline{\mathtt{out}}\langle f, h_f\rangle \big)$$

$$AG \stackrel{\text{def}}{=} a[\ \overline{\mathtt{in}}\langle a, h_a\rangle.\mathtt{open}\langle k\rangle.(x).x.\overline{\mathtt{open}}\langle a\rangle.Q\ ]$$

Note that here, unlike [9], the names f an a, of the firewall and agent respectively, can be considered public information; the security of the system resides in keeping the passwords h_f and h_a private.

We now turn our attention to some example laws which we can justify straightforwardly using bisimulations. In [9] it is shown that by establishing a set of basic set of such laws between ambients non-trivial reasoning can be carried out. Indeed most of our laws are taken directly from that paper, or are simple modifications thereof. Here we show how they can be established using bisimulations, rather than the more complicated contextual reasoning in [9].

The simplest example is

$$n[\ ] = \mathbf{0}$$

These two processes are bisimilar because the singleton set

$$\{(n[\mathbf{0}]\ ,\ \mathbf{0})\}$$

is a trivial bisimulation; neither side can perform any action.

Another example is the *perfect firewall equation* of [6]

$$(\boldsymbol{\nu} n)n[P] = \mathbf{0}$$

where $n \notin \mathrm{fn}(P)$. This law is not true in our setting, nor does it hold for the Safe Ambients of [9]. For example, consider the case when P is given by

$$P = \mathtt{in}\langle k\rangle.P'$$

with $k \neq n$ and $n \notin \mathrm{fn}(P')$. Then the context

$$C[\cdot] = [\cdot] \mid k[\ \overline{\mathtt{in}}\langle k\rangle.r[\mathtt{out}\langle k\rangle]\ ] \mid \overline{\mathtt{out}}\langle k\rangle$$

is capable of distinguishing the two processes. Indeed, when $r \notin \mathrm{fn}(P)$, we have $C[(\boldsymbol{\nu} n)n[P]]\Downarrow_r$ whereas $C[\mathbf{0}]\not\Downarrow_r$. Roughly, this means that the

movements of secret ambients are not visible in Mobile Ambients while they are in the presence of co-capabilities.

However in our setting we can prove a law similar to the perfect firewall equation:

THEOREM 6.1.

$$(\boldsymbol{\nu} n_1)(\boldsymbol{\nu} n_2) n_1[n_2[P]] \approx \mathbf{0}$$

Proof: Again

$$\{\big(\,(\boldsymbol{\nu} n_1)(\boldsymbol{\nu} n_2) n_1[n_2[P]]\,,\,\mathbf{0}\,\big)\}$$

is a bisimulation since neither side can perform any external action. □

Here are a collection of laws taken from [9]:

THEOREM 6.2.

1. $\boldsymbol{\nu} h(m[\mathtt{in}\langle n, h\rangle.P] \mid n[\overline{\mathtt{in}}\langle n, h\rangle.Q]) \approx \boldsymbol{\nu} h(n[Q \mid m[P]])$
2. $k[m[\mathtt{in}\langle n, h\rangle.P] \mid n[\overline{\mathtt{in}}\langle n, h\rangle.Q])] \approx k[n[Q \mid m[P]]]$
3. $\boldsymbol{\nu} h(\mathtt{open}\langle m, h\rangle.P \mid m[\overline{\mathtt{open}}\langle m, h\rangle.Q]) \approx \boldsymbol{\nu} h(P \mid Q)$
4. $k[\mathtt{open}\langle m, h\rangle.P \mid m[\overline{\mathtt{open}}\langle m, h\rangle.Q]] \approx k[P \mid Q]$
5. $\boldsymbol{\nu} h(n[m[\mathtt{out}\langle n, h\rangle.Q]] \mid \overline{\mathtt{out}}\langle n, h\rangle.P) \approx \boldsymbol{\nu} h(m[Q] \mid P)$
6. $n[\langle E\rangle.P \mid (x).Q] \approx n[P \mid Q\{^{E}/_{x}\}]$

Proof: By exhibiting the appropriate bisimulation. In all cases the bisimulation has a similar form:

$$\mathcal{S} = \{(LHS, RHS)\} \cup \approx$$

where LHS, RHS denote the left hand side, right hand side respectively of the identity. In the proof of part 5 we require the law $n[\,] \approx \mathbf{0}$. □

These laws may now be used to prove our version of crossing a firewall:

THEOREM 6.3. *If* $h_a \notin \mathrm{fn}(P)$ *and* $h_f \notin \mathrm{fn}(Q)$, *then:*

$$\boldsymbol{\nu} h_a(AG \mid FW) \approx \boldsymbol{\nu}(h_a h_f) f[P \mid Q]$$

Proof: Similar to the proof of Equation (15) of [9], but now applying Laws 5, 1, 4, 6, 1, 4 of Theorem 6.2. □

Note that because of the security of the system is only maintained by keeping the passwords secret, in this law we have to restrict on these, rather than on the names f and a.

7 Conclusion and Related Work

We have introduced the calculus SAP, a variant of Levi and Sangiorgi's Safe Ambients enriched with passwords. In SAP by managing passwords, for example generating new ones and distributing them selectively, an ambient may now program who may migrate into its computation space, and when. Moreover ambients in SAP may provide different services depending on the passwords exhibited by its clients to enter it.

The main result of the paper is

- an lts based operational semantics for SAP
- a bisimulation based equivalence over this lts which coincides with barbed congruence.

Higher-order ltss for Mobile Ambients can be found in [3, 20]. But we are not aware of any form of bisimilarity defined using these ltss. Our lts is inspired by that in [9] which differs from ours mainly for two reasons. The first is that in our lts the co-capability $\overline{\texttt{out}}$ is exercised by the target computation space and not by the surrounding ambient; this allows us to avoid the action $?n$ of [9] for which it is difficult to conceive of a distinguishing context. The second point is that we have a different kind of concretion with a different meaning. In SA a concretion $\boldsymbol{\nu}\tilde{p}\langle P\rangle Q$ means that P is moving whereas Q stays where it is; in SAP we are more precise, a concretion $\boldsymbol{\nu}\tilde{p}\langle P\rangle_n Q$ means that P is the computation inside ambient n and Q is the computation outside n. This allows us to define a reasonable lts and therefore use a standard notion of bisimilarity (c.f. Definitions 4.11 and 4.12) for SAP.

A simple first-order lts for MA without restriction is proposed by Sangiorgi in [17]. Using this lts the author defines an *intensional* bisimilarity for MA which separates terms on the basis of their internal structure. Sangiorgi shows that his bisimilarity coincides with the equivalence induced by the logic for MA given in [5] and and more surprisingly with structural congruence[1]. This result somehow shows that the algebraic theory of Mobile Ambient is quite poor.

Our lts can be smoothly adapted to MA and SA. We believe that in both cases it is possible to derive a bisimulation congruence similar to our ambient bisimilarity. However in both cases there are severe difficulties in proving that such bisimilarity completely characterise barbed congruence. In MA ambient movements are completely asynchronous (there are

[1] This is proved in synchronous MA where communication, like in SAP, is synchronous; in asynchronous MA the difference between bisimilarity and $\equiv$ is captured by a single axiom.

no co-capabilities) and this leads to a *stuttering* phenomena originated by ambients that may repeatedly enter and exit another ambient. As a consequence, it is far from trivial to find a distinguishing context for actions like $\mathtt{enter}\langle n\rangle$. Stuttering does not show up in SA and SAP because movements are achieved by means of synchronisation between a capability and a co-capability. However characterisations results for SA, similar to Theorem 4.17, are very difficult to prove. The technical problem is due to the difficulty in conceiving a distinguishing context for actions like $\overline{\mathtt{enter}}\langle n\rangle$. Roughly, in order to test that a process can allow entry to an ambient n a context has to move an ambient m into n. In SAP probing for this using fresh passwords ensures that ambient m has indeed been accepted at n. Without fresh passwords there would be no distinguishing feature of the particular ambient m which could be used in the probe. Alternatively, instead of using passwords, one may think of equipping SA [2] with *guarded choice* à la CCS. We believe that in SA with guarded choice ambient bisimilarity coincides with barbed congruence. The proof that ambient bisimilarity implies barbed congruence does not present particular difficulties. The interesting part is the converse where guarded choice plays a crucial role in the proof. However, a general implementation of guarded choice is problematic as it involves non-local consensus decisions. For this reason we prefer our version of ambients with passwords, SAP, which we believe is a good basis for developing interesting typing disciplines for mobile code making use of passwords. Even more, we think we can derive a labelled characterisation of typed barbed congruence along the lines of Hennessy and Rathke's [7].

Acknowledgements

The first author would like to thank Julian Rathke and Davide Sangiorgi for insightful discussions on higher-order process calculi and Safe Ambients, respectively.

References

[1] R. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous π-calculus. *Theoretical Computer Science*, 195:291–324, 1998.

[2] M. Bugliesi, G. Castagna, and S. Crafa. Boxed ambients. In *TACS'01 Proc. of the 4th Int. Conference on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science. Springer-Verlag, 2001. to appear.

[3] Luca Cardelli and Andrew Gordon. A commitment relation for the ambient calculus. Unpublished notes, 1996.

[2] More precisely, SA with our operational semantics for $\overline{\mathtt{out}}$.

[4] Luca Cardelli and Andrew D. Gordon. Types for mobile ambients. In *26th Annual Symposium on Principles of Programming Languages (POPL) (San Antonio, TX)*, pages 79–92. ACM, 1999.

[5] Luca Cardelli and Andrew D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POLP-00)*, pages 365–377, N.Y., January 19–21 2000. ACM Press.

[6] Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000. An extended abstract appeared in *Proceedings of FoSSaCS '98*: 140–155.

[7] M. Hennessy and J. Rathke. Typed behavioural equivalences for processes in the presence of subtyping. Computer Science Report 2/01, University of Sussex, 2001.

[8] K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 152(2):437–486, 1995.

[9] F. Levi and D. Sangiorgi. Controlling interference in ambients. Short version appeared in *Proc. 27th POPL*, ACM Press, 2000.

[10] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[11] R. Milner. The polyadic π-calculus: a tutorial. Technical Report ECS–LFCS–91–180, LFCS, Dept. of Comp. Sci., Edinburgh Univ., October 1991. Also in *Logic and Algebra of Specification*, ed. F.L. Bauer, W. Brauer and H. Schwichtenberg, Springer Verlag, 1993.

[12] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, (Parts I and II). *Information and Computation*, 100:1–77, 1992.

[13] R. Milner and D. Sangiorgi. Barbed bisimulation. In W. Kuich, editor, *Proc. 19th ICALP*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695. Springer Verlag, 1992.

[14] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis CST–99–93, Department of Computer Science, University of Edinburgh, 1992.

[15] D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. *Information and Computation*, 111(1):120–153, 1994.

[16] D. Sangiorgi. Bisimulation for Higher-Order Process Calculi. *Information and Computation*, 131(2):141–178, 1996.

[17] D. Sangiorgi. Extensionality and intensionality of the ambient logic. In *Proc. 28th POPL*. ACM Press, 2001.

[18] D. Sangiorgi and R. Milner. The problem of "Weak Bisimulation up to". In W.R. Cleveland, editor, *Proc. CONCUR '92*, volume 630 of *Lecture Notes in Computer Science*, pages 32–46. Springer Verlag, 1992.

[19] D. Sangiorgi and A. Valente. A distributed abstract machine for Safe Ambients. To appear in *Proc. 28th POPL*, 2001.

[20] Maria Grazia Vigliotti. Transitions systems for the ambient calculus. Master thesis, Imperial College of Science, Technology and Medicine (University of London), September 1999.

[21] Jan Vitek and Giuseppe Castagna. Seal: A framework for secure mobile computations. In *Internet Programming Languages*, 1999.

[22] W.P. Weijland. *Synchrony and Asynchrony in Process Algebra*. PhD thesis, University of Amsterdam, 1989.