Bisimulation Congruences in Safe Ambients

Massimo Merro and Matthew Hennessy
COGS, University of Sussex, UK

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.

Our notion of bisimulation is used 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 [5] 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, [11], using the construct $\boldsymbol{\nu} nP$; here knowledge of n is restricted to P. For example the system

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

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 $m[\dots]$, in addition to the capability $\mathtt{open}\langle k\rangle$, which allows the opening of any ambient named k which migrates into the computation space of n.

Papers such as [5, 3] 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 [5] 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*

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

This raises two questions:

- What is the appropriate notion of semantic equivalence $\approx$ for ambients?

- What proof methods exist for establishing such equivalences?

This is the topic of the current paper.

In [8] it has been argued that the calculus MA, as given in [5], is qualitatively different from more standard process calculi such as the Picalculus [11]. 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

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

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 [8]. 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, a password, 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 [7, 12] 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 i) is a congruence for the language, that is is preserved by all constructs of the language, ii) preserves, in some sense, the reduction semantics of the language iii) preserves *barbs*, that is preserves some simple observational property of terms. A formal definition is given in Definition 2.2. This relation has all of the extensional properties we require of semantic equivalence but it is very difficult to reason about; see for example the proof of the equational laws in [8]. However bisimulation relations, because of their co-inductive nature, provide powerful proof techniques for establishing equivalences, [13, 17, 14]; these are based on descriptions of processes in terms of 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 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. The lts contains, as expected, actions for all of the capabilities

Table 1 The Calculus SAP

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

Processes:

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

Capabilities:

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

and co-capabilities in the language[1]. These take the form $P \xrightarrow{\alpha} Q$, a typical example being

$$\mathtt{in}\langle n\rangle.P \xrightarrow{\mathtt{in}\langle n\rangle} P$$

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 i) which ambient enters n and ii) what residual code remains behind. Such actions will have the form

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

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.

In order to obtain our lts based characterisation of $\cong$ we use these higher-order actions to define a version of *weak moves* between processes. These weak moves are defined in two steps. The first replaces actions whose residuals are concretions with actions whose residuals are simple processes. For example, the $\mathtt{enter}\langle n\rangle$ action above is replaced by the family of moves

$$P \xrightarrow{\mathtt{enter}\langle n\rangle R} \boldsymbol{\nu}\tilde{m}\, n[A \mid R] \mid P'.$$

Thus weak moves $\stackrel{\alpha}{\Longrightarrow}$ are defined in the standard manner as $\xrightarrow{\tau}^{*} \xrightarrow{\alpha} \xrightarrow{\tau}^{*}$.

The main result of the paper is that, in SAP, the resulting (weak) bisimulation equivalence $\approx$, based on these weak moves, 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, as in [5, 8]. 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 extended abstract ends with Section 7, containing a discussion of our results and a comparison with related work. In this version of the paper proofs are omitted or just sketched. Complete proofs can be found in the full version [9], available at `http://www.cogs.susx.ac.uk/reports.html`.

2 The Calculus SAP

The syntax of processes is given in Table 1 and is basically the same as that in [5], except that each of the original capabilities has a co-capability, as in [8], and that now each capability has an extra argument h,

[1] Here, as in much of the paper, we will ignore passwords unless they play a central role in the discussion

which may be looked upon as a *password*. The calculus has replicated prefixing, rather than full replication, or recursion; this results in an image-finite labelled transition system. Finally for simplicity we have omitted communication; this will be added in Section 5.

We 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.

As in the Picalculus the reduction semantics is based on an auxiliary relation called *structural congruence* which brings the participants of a potential interaction into contiguous positions. The full definitions of structural congruence, $\equiv$, and the reduction relation, $\rightarrow$, can be found in the Appendix in Table 4. Both definitions are similar to those for SA, except for the passwords and the following crucial rule for *emigration*:

$$\text{(Red Out)} \quad m[\,n[\mathtt{out}\langle m,h\rangle.P \mid Q] \mid R\,] \;\mid\; \overline{\mathtt{out}}\langle m,h\rangle.S \;\;\rightarrow\;\; n[P \mid Q] \;\mid\; m[R] \;\mid\; S$$

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 [8] 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. As usual, we write $\Rightarrow$ to denote the reflexive and transitive closure of $\rightarrow$.

We end this section with the definition of what we believe to be an appropriate behavioural equivalence in SAP: *barbed congruence* [12, 7], based on a notion of observation. In ambients the observation predicate $P\downarrow_n$ is used to denote the possibility of process P of interacting with the environment via the ambient n; in [5] this is true whenever $P \equiv \boldsymbol{\nu}\tilde{m}(n[P_1] \mid P_2)$ where $n \notin \{\tilde{m}\}$. This is a reasonable definition of observation for MA as no authorisation is required to cross a boundary, and the presence of an ambient n at top level denotes a potential interaction between the process and the environment via n. However in SA, [8], 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 [8] $P\downarrow_n$ 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 names h, $\tilde{m}$, and processes 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 $P'\downarrow_n$.*

Notice that our notion of barb is simpler than that in [8]. Note also 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).

A relation $\mathcal{R}$ is i) *reduction closed* if $P\,\mathcal{R}\,Q$ and $P \rightarrow P'$ implies the existence of some Q' such that $Q \Rightarrow Q'$ and $P'\,\mathcal{R}\,Q'$; ii) *barb preserving* if $P\,\mathcal{R}\,Q$ and $P\downarrow_n$ implies $Q\Downarrow_n$.

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

Our choice of observation here may feel arbitrary. But in Section 4.1 we will show that $\cong$ remains invariant under a large choice of possible observation predicates.

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 $C.P \stackrel{C}{\longmapsto} Q$. 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 and therefore we would have the identity $n[P] \approx_{bad} \mathbf{0}$ regardless of P.

Table 2 Labels, Concretions, and Outcomes

$$\textit{Prefixes:}\quad \mu ::= \mathtt{in}\langle n,h\rangle \mid \mathtt{out}\langle n,h\rangle \mid \mathtt{open}\langle n,h\rangle \mid \overline{\mathtt{in}}\langle n,h\rangle \mid \overline{\mathtt{out}}\langle n,h\rangle \mid \overline{\mathtt{open}}\langle n,h\rangle$$

$$\textit{Concretions:}\quad K ::= \boldsymbol{\nu}\tilde{m}\langle P\rangle_n Q$$

$$\textit{Actions:}\quad \alpha ::= \tau \mid \mu \mid \mathtt{enter}\langle n,h\rangle \mid \overline{\mathtt{enter}}\langle n,h\rangle \mid \mathtt{exit}\langle n,h\rangle \mid \mathtt{pop}\langle n,h\rangle \mid \mathtt{free}\langle n,h\rangle$$

$$\textit{Outcomes:}\quad O ::= P \mid K$$

Table 3 Labelled Transition System

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

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

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

$$(\tau\ \text{Out})\ \frac{P \xrightarrow{\mathtt{pop}\langle n,h\rangle} P' \quad Q \xrightarrow{\overline{\mathtt{out}}\langle n,h\rangle} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad (\tau\ \text{In})\ \frac{P \xrightarrow{\mathtt{enter}\langle n,h\rangle} \boldsymbol{\nu}\tilde{p}\langle P_1\rangle_n P_2 \quad Q \xrightarrow{\overline{\mathtt{enter}}\langle n,h\rangle} \boldsymbol{\nu}\tilde{q}\langle Q_1\rangle_n Q_2^{(*)}}{P \mid Q \xrightarrow{\tau} \boldsymbol{\nu}\tilde{p}\boldsymbol{\nu}\tilde{q}(n[P_1 \mid Q_1] \mid P_2 \mid Q_2)}$$

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

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

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

(*) In rule (τ In) we require $((\mathrm{fn}(P_1) \cup \mathrm{fn}(P_2)) \cap \{\tilde{q}\}) = ((\mathrm{fn}(Q_1) \cup \mathrm{fn}(Q_2)) \cap \{\tilde{p}\}) = \emptyset$

However the actions above can be considered the basis of further capabilities. For example in the system $n[\mathtt{in}\langle m,h\rangle.P] \mid Q$ there is the capability to *enter* ambient m with password h. 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 $P \xrightarrow{\alpha} O$ where the range of α and of O, the *outcomes*, are given in Table 2. 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 [8], are given in Table 3. Here we only explain the rules for *emigration*. A full explanation of the lts can be found in [9].

The driving force behind the emigration is the activation of the prefix $\mathtt{out}\langle n,h\rangle$. It induces a capability in an ambient m to emigrate from n with password h, which we formalise as a new action $\mathtt{exit}\langle n,h\rangle$. Thus, for $k \notin \{n,h\}$, an application of the rule (Exit), followed by (Par Exit) and (Res) gives

$$\boldsymbol{\nu} k\big(m[\,\mathtt{out}\langle n,h\rangle.P_1\,] \;\mid\; P_2\,\big) \xrightarrow{\mathtt{exit}\langle n,h\rangle} \boldsymbol{\nu} k\big(\langle P_2\rangle_n m[P_1]\big)$$

Here when this capability is exercised the code P_2 will remain inside the ambient n while the ambient $m[P_1]$ will move outside. The structural rule (Res) 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 $\boldsymbol{\nu} r O$ is a shorthand for $\boldsymbol{\nu}\tilde{m}\langle P\rangle_n \boldsymbol{\nu} r Q$, 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 $\mathrm{fn}(R) \cap \{\tilde{m}\} = \emptyset$.

However, in the example above, to actually effect the emigration of m we need a further context, namely the ambient n from which to emigrate. This leads to another action, called $\mathtt{pop}\langle n,h\rangle$, with the associated rule (Pop); an application of which gives:

$$n[\boldsymbol{\nu} k\,(m[\mathtt{out}\langle n,h\rangle.P_1] \;\mid\; P_2)\,] \xrightarrow{\mathtt{pop}\langle n,h\rangle} \boldsymbol{\nu} k\big(\,n[P_2] \;\mid\; m[P_1]\,\big)$$

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

$$n[\boldsymbol{\nu} k\,(m[\mathtt{out}\langle n,h\rangle.P_1] \;\mid\; P_2)\,] \;\mid\; \overline{\mathtt{out}}\langle n,h\rangle.Q \;\xrightarrow{\tau}\; \boldsymbol{\nu} k\big(\,n[P_2] \;\mid\; m[P_1]\,\big) \;\mid\; Q$$

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

Theorem 3.1 *If $P \xrightarrow{\tau} P'$ then $P \to P'$. If $P \to P'$ then $P \xrightarrow{\tau}\equiv P'$.*

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 co-inductive characterisation of $\cong$.

4.1 Barbs

According to [5, 8], the predicate $P{\downarrow_n}$ (cf. Definition 2.1) 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 our 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 $P \xrightarrow{\mathtt{free}\langle n,h\rangle} P'$ for some h and P'.*

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 2.

Definition 4.2 *We write $P{\downarrow_\alpha}$ if $P \xrightarrow{\alpha}$. We write $P{\Downarrow_\alpha}$ if $P \Rightarrow\xrightarrow{\alpha}$.*

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 *and preserves λ barbs.*

It is very easy to establish that if $P \cong_\lambda Q$ then i) $P{\Downarrow_{\lambda\langle n,h\rangle}}$ iff $Q{\Downarrow_{\lambda\langle n,h\rangle}}$; ii) $P \Rightarrow P'$ 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 for any λ in $\mathcal{L}$ it holds that $P \cong Q$ iff $P \cong_\lambda Q$.*

Proof: [Sketch] Since the $\cong$ and $\cong_\lambda$ differ only in the barb which is used it suffices to show $\downarrow_n$ and $\downarrow_{\lambda\langle n,h\rangle}$ imply each other. We only examine the case $\lambda = \overline{\texttt{enter}}$. The other cases are similar.
Let us consider first the implication from left to right. Let $P \cong Q$ and $P\downarrow_{\overline{\texttt{enter}}\langle n,h\rangle}$; we want to conclude that $Q\Downarrow_{\overline{\texttt{enter}}\langle n,h\rangle}$. Consider the context $S_1[\cdot] = [\cdot] \mid f[\texttt{in}\langle n,h\rangle.\texttt{out}\langle n,k\rangle] \mid \overline{\texttt{out}}\langle n,k\rangle.g[\overline{\texttt{open}}\langle g\rangle]$. If f,g and k are fresh to R, then the context $S_1[\cdot]$ has the property that $R\Downarrow_{\overline{\texttt{enter}}\langle n,h\rangle}$ iff $S_1[R]\Downarrow_g$. For $P \cong Q$ implies $S_1[P] \cong S_1[Q]$ which in turn implies $S_1[Q]\Downarrow_g$, from which we have the required $Q\Downarrow_{\overline{\texttt{enter}}\langle n,h\rangle}$.
As to the implication from right to left, let $P \cong_{\overline{\text{enter}}} Q$ and $P\downarrow_n$, then we want to conclude that $Q\Downarrow_n$. By Lemma 4.1, if $P\downarrow_n$ then there exists h such that P performs an action $\texttt{free}\langle n,h\rangle$. Thus, we define a context: $S_2^h[\cdot] = [\cdot] \mid \texttt{open}\langle n,h\rangle.g[\overline{\texttt{in}}\langle g\rangle]$. If g is fresh to R, then the context $S_2^h[\cdot]$ has the required properties that: i) $S_2^h[R]\Downarrow_{\overline{\texttt{enter}}\langle g\rangle}$ implies $R\Downarrow_n$; ii) $R\Downarrow_n$ implies $\exists h.\ S_2^h[R]\Downarrow_{\overline{\texttt{enter}}\langle g\rangle}$. This is sufficient to establish $Q\Downarrow_n$. $\Box$

In the proof above, the use of the fresh password k in the definition of $S_1[\cdot]$ is essential. Note also that the case $\lambda = \overline{\texttt{enter}}$ shows that the Levi and Sangiorgi's definition of barb, [8], can be simplified, to coincide with our original definition.

4.2 Labelled Bisimilarity

One possible approach to defining a behavioural equivalence would be to adapt to our language SAP the notion of *higher-order weak bisimilarity* given in [15] for HOπ. This uses weak actions of the form $\to^* \xrightarrow{\alpha}$, and since certain actions have concretions as residuals it also requires a method for comparing concretions.

In the full version of the paper we show that the resulting equivalence, which we call delay bisimilarity, is strictly contained in barbed congruence. This is not surprising since it uses weak actions which do not permit τ-moves after visible actions.

The co-inductive characterisation of barbed congruence presented here is based on an extension of the lts of Table 3 in which the use of concretions is eliminated. We do this by "applying" them to arbitrary processes:

Definition 4.5 $\boldsymbol{\nu}\tilde{m}\langle P\rangle_n Q \bullet R \stackrel{\text{def}}{=} \boldsymbol{\nu}\tilde{m}(n[P \mid R] \mid Q)$ *where* $\{\tilde{m}\}$ *is chosen so that* $\{\tilde{m}\} \cap \text{fn}(R) = \emptyset$.

With this form of application we can now replace the higher-order actions which have concretions as residuals, $\texttt{enter}\langle n,h\rangle$, $\texttt{exit}\langle n,h\rangle$, and $\overline{\texttt{enter}}\langle n,h\rangle$, with the family of actions $\texttt{enter}\langle n,h\rangle R$, $\texttt{exit}\langle n,h\rangle R$, $\overline{\texttt{enter}}\langle n,h\rangle m[R]$, respectively, whose residuals will be processes; here R and $m[R]$ represent part of the contribution of the environment to the performance of the higher-order actions.

Definition 4.6 (Ambient Transitions) *Let m be an arbitrary name and R an arbitrary process. Then:*

- $P \xrightarrow{\alpha R} K \bullet R$ *if* $P \xrightarrow{\alpha} K$ *for* $\alpha \in \{\texttt{enter}\langle n,h\rangle, \texttt{exit}\langle n,h\rangle\}$
- $P \xrightarrow{\overline{\texttt{enter}}\langle n,h\rangle m[R]} K \bullet m[R]$ *if* $P \xrightarrow{\overline{\texttt{enter}}\langle n,h\rangle} K$

The ambient transitions are defined from processes to processes and therefore give rise to *weak* transitions of a standard form: i) $\stackrel{\alpha}{\Longrightarrow}$ denotes $\xrightarrow{\tau}^* \xrightarrow{\alpha} \xrightarrow{\tau}^*$; ii) $\stackrel{\hat{\alpha}}{\Longrightarrow}$ denotes $\xrightarrow{\tau}^*$ if $\alpha = \tau$ and $\stackrel{\alpha}{\Longrightarrow}$ otherwise.

Definition 4.7 (Ambient Bisimilarity) *A symmetric relation $\mathcal{S}$ is an* ambient bisimulation *if $P\ \mathcal{S}\ Q$ and $P \xrightarrow{\alpha} P'$ 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 $\mathcal{S}$.*

Notice that the bisimilarity above considers only actions from processes to processes.

Theorem 4.8 *Ambient bisimilarity is a congruence.*

Proof: [Sketch] Let $\mathcal{S}$ be the least equivalence relation which i) contains the relation $\approx$, and ii) is preserved by restriction, parallel composition, and ambient operators. We prove that $\mathcal{S}$ is an ambient bisimilarity up to $\equiv$, by induction on the definition of $\mathcal{S}$.
We only consider the case when $P \mid R\ \mathcal{S}\ Q \mid R$ because $P\ \mathcal{S}\ Q$. Then, we carry out an inductive analysis on the transition $P \mid R \xrightarrow{\alpha} O$. The most interesting case is when $\alpha = \tau$. Let us consider the case when

$P \mid R \xrightarrow{\tau} O$ because $P \xrightarrow{\overline{\mathtt{enter}}\langle n,h\rangle} K_1$ and $R \xrightarrow{\mathtt{enter}\langle n,h\rangle} \boldsymbol{\nu}\tilde{r}\langle m[R_1]\rangle_n R_2$, with $O \equiv \boldsymbol{\nu}\tilde{r}\big((K_1 \bullet m[R_1]) \mid R_2\big)$. By definition, $P \xrightarrow{\overline{\mathtt{enter}}\langle n,h\rangle m[R_1]} K_1 \bullet m[R_1] = P'$. By the induction hypothesis, $P \;\mathcal{S}\; Q$ and so there is Q' such that $Q \Rightarrow \xrightarrow{\overline{\mathtt{enter}}\langle n,h\rangle m[R_1]} K_2 \bullet m[R_1] \Rightarrow\equiv Q'$ with $P' \;\mathcal{S}\; Q'$. Thus, $P \mid R \xrightarrow{\tau} O \equiv \boldsymbol{\nu}\tilde{r}\big(P' \mid R_2\big)$ and $Q \mid R \Rightarrow \xrightarrow{\tau} \boldsymbol{\nu}\tilde{r}\big((K_2 \bullet m[R_1]) \mid R_2\big) \Rightarrow\equiv \boldsymbol{\nu}\tilde{r}\big(Q' \mid R_2\big) = O'$. 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. $\Box$

By Theorem 4.8 and Lemma 4.1, we can now conclude that ambient bisimilarity $\approx$ is contained in barbed congruence $\cong$. The next step is to prove that ambient bisimilarity completely characterises barbed congruence.

Theorem 4.9 *Ambient bisimilarity and barbed congruence coincide.*

Proof: [Sketch] By Theorem 4.8 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 $\equiv$. We just show one of the most interesting cases, that is when $\alpha = \overline{\mathtt{enter}}\langle n,h\rangle m[R]$.
Let $P \xrightarrow{\overline{\mathtt{enter}}\langle n,h\rangle m[R]} P'$, that is $P \xrightarrow{\overline{\mathtt{enter}}\langle n,h\rangle} K_1 = \boldsymbol{\nu}\tilde{p}\langle P_1\rangle_n P_2$, 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 $P' \;\mathcal{S}\; Q'$. We define:

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

where h_i are fresh names and $\oplus$ denotes internal choice. As $P \cong_{\mathrm{pop}} Q$ it follows that $C_{\alpha m[R]}[P] \cong_{\mathrm{pop}} C_{\alpha m[R]}[Q]$. So, if $C_{\alpha m[R]}[P] \xrightarrow{\tau}\xrightarrow{\tau} C_1[m[R \mid \mathtt{out}\langle n,h_1\rangle]]$, then there is a process Z such that $C_{\alpha m[R]}[Q] \Rightarrow Z$ and $C_1[m[R \mid \mathtt{out}\langle n,h_1\rangle]] \cong_{\mathrm{pop}} Z$. As a consequence, $Z\Downarrow_{\mathtt{pop}\langle n,h_1\rangle}$ and $Z\not\Downarrow_{\mathtt{pop}\langle n,h_2\rangle}$. 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, one can prove that there exist static contexts (in which the hole is not underneath prefixing or replication) $C'[\cdot], C''[\cdot]$ and $C_2[\cdot]$ such that:

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

Since h_1 is fresh and contexts $C_1[\cdot]$ and $C_2[\cdot]$ are static, $C_1[m[R \mid \mathtt{out}\langle n,h_1\rangle]] \cong_{\mathrm{pop}} Z = C_2[\mathtt{out}\langle n,h_1\rangle]$ implies $C_1[m[R]] \cong_{\mathrm{pop}} C_2[\mathbf{0}]$. We now have the required Q', namely $C_2[\mathbf{0}]$. An analysis of the above reductions gives $Q \Rightarrow \xrightarrow{\overline{\mathtt{enter}}\langle n,h\rangle m[R]} C'[m[R]]$, and again, since h_1 is fresh, one can easily prove that $C'[m[R]] \Rightarrow\equiv Q'$. We therefore have the required move $Q \xRightarrow{\overline{\mathtt{enter}}\langle n,h\rangle m[R]} Q'$.
$\Box$

We believe that the distinguishing contexts in the proof above can be defined without the use of passwords, except when α is an $\overline{\mathtt{enter}}$ 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. Moreover our rules for $\overline{\mathtt{out}}$, different from those in [8], play a crucial role in the distinguishing contexts for both $\overline{\mathtt{enter}}$ and $\overline{\mathtt{in}}$. The alternative semantics for $\overline{\mathtt{out}}\langle n\rangle$ given in [8] uses an auxiliary action $?n$ for which it is difficult to conceive of a distinguishing context.

5 Adding Communication

In this section we extend SAP to allow local communication inside ambients. The basic idea is to have an *output process* such as $\langle E\rangle.P$, which outputs the message E and then continues as P, and an input process

$(x).Q$ which on receiving a message binds it to x in Q which then executes; here occurrences of x in Q are bound. Notice that we have synchronous output; as discussed in [19, 16, 1] this is not unrealistic because communication is always local. The syntax of our extended language is given in the Appendix in Table 5.

The operational semantics is defined over processes, i.e. terms which have no free occurrences of variables, by introducing two new labels: (E) for input, $\langle - \rangle$ for output, and a new form of concretion $\boldsymbol{\nu}\tilde{p}\langle E \rangle Q$.

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

An input and an output process may interact as follows:

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

In Table 6 of the Appendix we give all the defining rules which should be added to those of Table 3 and Definition 4.6 to obtain the lts $P \xrightarrow{\mu} O$ 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 3 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 $P \xrightarrow{\langle - \rangle} \boldsymbol{\nu}\tilde{p}\langle E \rangle Q$ into a transition $P \xrightarrow{L} P'$ for some process P' and label L.

Let R be any term such that x is the only free variable in R; intuitively here x represents the placeholder for the message E which will be received via an output action $P \xrightarrow{\langle - \rangle} \boldsymbol{\nu}\tilde{p}\langle E \rangle Q$. Then, we define

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

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

We can now define the extra ambient transition, $\xrightarrow{\langle - \rangle R}$, for any such R to add to those in Definition 4.6. Let

$$P \xrightarrow{\langle - \rangle R} K \bullet R \ \text{ if } \ P \xrightarrow{\langle - \rangle} K.$$

The above transitions can be used to generalise, in the standard manner, the definition of ambient bisimilarity, $\approx$, to arbitrary terms of our extended message-passing language. For any two arbitrary terms T, U we write $T \approx U$ if for all substitutions σ, mappings from variables to names, we have $T\sigma \approx U\sigma$.

Theorem 5.1 *Relations $\cong$ and $\approx$ coincide over arbitrary terms in the message-passing language.*

In [9] we point out how a simpler, and perhaps more natural, rule for output processes which avoids concretions would make the ambient bisimilarity strictly included in 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 [8] 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 analysed using our bisimulations.

In [9] we show how passwords can be used to rewrite the protocol to route packets to various destinations given in [5, 8]. Here we focus on the protocol for controlling accesses through a firewall [5, 8]. Our version is inspired by that in [8] 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.

$$
\begin{aligned}
FW &\stackrel{\text{def}}{=} \boldsymbol{\nu} h_f\big(\, f[\,\overline{\texttt{in}}\langle f,h_f\rangle.\texttt{open}\langle a\rangle.P \;\mid\; k[\texttt{out}\langle f,h_f\rangle.\texttt{in}\langle a,h_a\rangle.\overline{\texttt{open}}\langle k\rangle.\langle \texttt{in}\langle f,h_f\rangle\rangle]\,] \;\mid\; \overline{\texttt{out}}\langle f,h_f\rangle\,\big)\\
AG &\stackrel{\text{def}}{=} a[\,\overline{\texttt{in}}\langle a,h_a\rangle.\texttt{open}\langle k\rangle.(x).x.\overline{\texttt{open}}\langle a\rangle.Q\,]
\end{aligned}
$$

Note that here, unlike [8], the names f and 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 [8] it is shown that by establishing a set of basic 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 [8].

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. Note that this law is not true in MA.

An important law in MA, is the *perfect firewall equation* $(\boldsymbol{\nu} n)n[P] = \mathbf{0}$ where $n \notin \text{fn}(P)$. This law is not true in our setting, nor does it hold in SA. For example, consider the case when P is given by $P = \texttt{in}\langle k\rangle.P'$ with $k \neq n$ and $n \notin \text{fn}(P')$. Then the context $C[\cdot] = [\cdot] \mid k[\,\overline{\texttt{in}}\langle k\rangle.r[\texttt{out}\langle k\rangle]\,] \mid \overline{\texttt{out}}\langle k\rangle$ is capable of distinguishing the two processes. 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: $\{(\,(\boldsymbol{\nu} n_1)(\boldsymbol{\nu} n_2)n_1[n_2[P]]\ ,\ \mathbf{0}\,)\}$ is a bisimulation since neither side can perform any external action. □

Here are a collection of laws taken from [8]:

Theorem 6.2

1. $\boldsymbol{\nu} h(m[\texttt{in}\langle n,h\rangle.P] \mid n[\overline{\texttt{in}}\langle n,h\rangle.Q]) \approx \boldsymbol{\nu} h(n[Q \mid m[P]])$
2. $k[m[\texttt{in}\langle n,h\rangle.P] \mid n[\overline{\texttt{in}}\langle n,h\rangle.Q]] \approx k[n[Q \mid m[P]]]$
3. $\boldsymbol{\nu} h(\texttt{open}\langle m,h\rangle.P \mid m[\overline{\texttt{open}}\langle m,h\rangle.Q]) \approx \boldsymbol{\nu} h(P \mid Q)$
4. $k[\texttt{open}\langle m,h\rangle.P \mid m[\overline{\texttt{open}}\langle m,h\rangle.Q]] \approx k[P \mid Q]$
5. $\boldsymbol{\nu} h(n[m[\texttt{out}\langle n,h\rangle.Q]] \mid \overline{\texttt{out}}\langle n,h\rangle.P) \approx \boldsymbol{\nu} h(m[Q] \mid P)$
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 and very simple form:

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

where LHS, RHS denote the left hand side, right hand side respectively of the identity. □

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

Theorem 6.3 *If $h_a \notin \text{fn}(P)$ and $h_f \notin \text{fn}(Q)$, then $\boldsymbol{\nu} h_a(AG \mid FW) \approx \boldsymbol{\nu}(h_a h_f)f[P \mid Q]$.*

Proof: Similar to the proof in [8], but now applying Laws 5, 1, 4, 6, 1, 4 of Theorem 6.2. □

Note that, because 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

Higher-order ltss for Mobile Ambients can be found in [2, 18]. However we are not aware of any form of bisimilarity defined using these ltss. Our lts is inspired by that in [8] which differs from ours mainly in two respects. 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 [8] 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 4.6 and 4.7) for SAP.

A simple first-order lts for MA without restriction operator is proposed by Sangiorgi in [16]. 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 [4] and and more surprisingly with structural congruence[2]. This result somehow shows that the algebraic theory of Mobile Ambient is quite poor.

With some work our lts can be adapted to both 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 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 $\texttt{enter}\langle n\rangle$. 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 4.9, are very difficult to prove. The technical problem is due to the difficulty in conceiving a distinguishing context for actions like $\overline{\texttt{enter}}\langle n\rangle$. Roughly, in order to test that a process can allow entry to an ambient n, a distinguishing 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 [3] 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 [6].

Acknowledgements

The authors would like to thank Julian Rathke and Davide Sangiorgi for insightful discussions on higher-order process calculi and Safe Ambients, respectively, and the anonymous referees for detailed comments. Research funded by EPSRC grant GR/M71169

References

[1] 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.

[2] 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.

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

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

[3] 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.

[4] 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.

[5] 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.

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

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

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

[9] M. Merro and M. Hennessy. Bisimulation congruences in safe ambients. Computer Science Report 5/01, University of Sussex, 2001.

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

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

[12] 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.

[13] 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.

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

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

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

[17] 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.

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

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

A Appendix

Table 4 Structural Congruence and Reduction Rules

$P \mid Q \equiv Q \mid P$	(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)

$\equiv$ is the least equivalence relation which i) satisfies the axioms and rules above and
ii) is preserved by all operators except prefixing and replication.

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

$\rightarrow$ is the least equivalence relation which i) satisfies the rules above and
ii) is preserved by all operators except prefixing and replication.

Table 5 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
$\mid$	$\mathtt{out}\langle n, h\rangle$	may exit out of n
$\mid$	$\mathtt{open}\langle n, h\rangle$	may open n
$\mid$	$\overline{\mathtt{in}}\langle n, h\rangle$	allow enter
$\mid$	$\overline{\mathtt{out}}\langle n, h\rangle$	allow exit
$\mid$	$\overline{\mathtt{open}}\langle n, h\rangle$	allow open

Expressions:

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

Guards:

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

Processes:

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

Concretions:

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

Table 6 Labelled Transition System - Communication

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

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

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