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 e ample 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 e hibited by its incoming clients.

We give an *lts* based operational semantics for SAP and a

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, $\mathtt{o}\ \mathtt{t}\langle n\rangle$, in which case the system will have three concurrent ambients:

$$k[\ldots] \quad | \quad n[\,\mathtt{open}\langle k\rangle.P\,] \quad | \quad m[\ {}_1 \mid {}_2\,]$$

Papers such as [6, 4] demonstrate that this calculus is very effective in formally describing the

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

$$m[\,\mathtt{in}\langle n\rangle.\ \ _1 \mid\ \ _2\,] \ \mid\ n[P]$$

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

$$n[\,m[\ \ _1 \mid\ \ _2\,] \mid P_1 \mid P_2\,]$$

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

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

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

if the action $\mathtt{enter}\langle n\rangle$ is performed the migrating ambient is $m[P]$, while the residual code is $\ \ |$. 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

$$P \vdash^{\mathtt{enter}\langle n\rangle}$$

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
	$\mathtt{o\ t}\langle n, h\rangle$	may exit out of n
	$\mathtt{open}\langle n, h\rangle$	may open n
	$\overline{\mathtt{in}}\langle n, h\rangle$	allow enter
	$\overline{\mathtt{o\ t}}\langle n, h\rangle$	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 $P \xrightarrow{\alpha}$, 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

$P \mid \quad \equiv \quad \mid P$	(Struct Par Comm)
$(P \mid \quad) \mid R \equiv P \mid (\quad \mid R)$	(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 $\boldsymbol{\nu} n(P \mid \quad) \equiv P \mid \boldsymbol{\nu} n$	(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:

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

TABLE 3 Reduction Rules

$n[\mathtt{in}\langle\ \ ,h\rangle.P \mid Q] \mid\ \ [\overline{\mathtt{in}}\langle\ \ ,h\rangle.R \mid\ \ ] \rightarrow\ \ [\,n[P \mid Q] \mid R \mid\ \ ]$	(Red In)
$[\,n[\mathtt{out}\langle\ \ ,h\rangle.P \mid Q] \mid R\,] \mid \overline{\mathtt{out}}\langle\ \ ,h\rangle.\ \ \rightarrow n[P \mid Q] \mid\ \ [R] \mid$	(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\ \ $ implies $P \rightarrow$	(Red Str)

which intuitively means that P can evolve to 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 *exerciTamaxTfTRinaxtoomTdTdpat*

cates. In [6] the predicate $P \downarrow_n$ is used to denote the possibility of process P of

TABLE 4 Labels, Concretions, and Outcomes

Actions:	μ	::=	$\mathtt{in}\langle n,h\rangle$	$\mathtt{o\ t}\langle n,h\rangle$	$\mathtt{open}\langle n,h\rangle$
			$\overline{\mathtt{in}}\langle n,h\rangle$	$\overline{\mathtt{o\ t}}\langle n,h\rangle$	$\overline{\mathtt{open}}\langle n,h\rangle$
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	::=	$\boldsymbol{\nu}\tilde{m}\langle P\rangle_n$		
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 $n[\mathtt{in}\langle m\rangle P.] \mid$ 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 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 , if for example α is a prefix from the language, or a *concretion*, of the form

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

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

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

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

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

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

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

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

follows:

$$m[\mathtt{in}\langle n\rangle.P_1] \mid P_2 \mid n[\overline{\mathtt{in}}\langle n\rangle.\ \ _1] \mid\ \ _2 \rightarrow n[\, m[P_1] \mid\ \ _1\,] \mid P_2 \mid\ \ _2$$

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

$$n[\overline{\mathtt{in}}\langle n\rangle.\ \ _1] \mid\ \ _2 \overset{\overline{\mathtt{enter}}\langle n\rangle}{\longmapsto} \langle\ \ _1\rangle_n\ \ _2$$

Here, after the action, $\ _1$ remains inside n, while $\ _2$ 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:

$$m[\mathtt{in}\langle n\rangle.P_1] \mid P_2 \mid n[\overline{\mathtt{in}}\langle n\rangle. \quad _1] \mid \quad _2 \stackrel{\tau}{\longmapsto} n[\, m[P_1] \mid \quad _1\,] \mid P_2 \mid \quad _2$$

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 $\boldsymbol{\nu}\tilde{m}\langle P\rangle_n \quad$, then $\boldsymbol{\nu}rO$ is a shorthand for $\boldsymbol{\nu}\tilde{m}\langle P\rangle_n\boldsymbol{\nu}r \quad$, if $r \notin \mathrm{fn}(P)$, and the concretion $\boldsymbol{\nu}(r\tilde{m})\langle P\rangle_n$

- $f\,P \vdash \xrightarrow{\texttt{enter}\langle n,h\rangle} \boldsymbol{\nu}\tilde{m}\langle P_1\rangle_n P_2$ *then* P_1 *is an ambient*
- $f\,P \vdash \xrightarrow{\texttt{exit}\langle n,h\rangle} \boldsymbol{\nu}\tilde{m}\langle P_1\rangle_n P_2$ *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. $f\,P \vdash \xrightarrow{\tau} P'$ *then* $P \to P'$.
2. $f\,P \to P'$ *then* $P \vdash \xrightarrow{\tau} \equiv P'$.

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 $P \vdash \xrightarrow{\alpha} O$. For instance,

LEMMA 4.1. *$P{\downarrow_n}$ iff $P \vdash \xrightarrow{\texttt{free}\langle n,h\rangle} P'$ for some h and P'.*

Proof: Straightforward. □

In this subsection, we prove that for all possible labels generated in our lts thein

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

$$S_2^h[\cdot] \stackrel{\text{d\_f}}{=} [\cdot] \mid \texttt{open}\langle n,h\rangle.k[r[\overline{\texttt{o t}}\langle k\rangle]]$$

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

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

This is sufficient to establish $\Downarrow_n$.

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

$$S_1[\cdot] \stackrel{\text{d\_f}}{=} [\cdot] \mid f[\texttt{in}\langle n,h\rangle.\texttt{o t}\langle n,k\rangle] \mid \overline{\texttt{o t}}\langle n,k\rangle.g[\overline{\texttt{open}}\langle g\rangle]$$

This context has the property that

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

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

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

This has the required property that:

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

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 $\lambda = \overline{\texttt{enter}}$ the use of the fresh password k in the definition of $S_1[\cdot]$ is essential. Note also thattocase

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

Since some actions result in concretions

$$- \;\; P \;\; | \;\; R \;\; \vdash^{\tau}$$

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

As $P \ \mathcal{S}\ $, we may use induction to find a $'$ such that $\stackrel{\mathtt{in}\langle m,h\rangle}{\longmapsto}$ $'$ and $P' \ \mathcal{S}\ '$. We may now let $K_2 = \langle\ '\rangle_n \mathbf{0}$ as $n[\ ] \stackrel{\mathtt{enter}\langle m,h\rangle}{\Longmapsto} K_2$ and since $\mathcal{S}$ is preserved by parallel composition and ambient constructor, we get $K_1 \bullet W = n[P' \mid W] \ \mathcal{S}\ n[\ ' \mid W] = K_2 \bullet W$, 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}\ $, by induction it holds that for all processes W there exists $K_2 = \boldsymbol{\nu}\tilde{q}\langle\ _1\rangle_n\ _2$ such that $\stackrel{\mathtt{exit}\langle n,h\rangle}{\Longmapsto} K_2$ and $K_1 \bullet W \ \mathcal{S}\ K_2 \bullet W$. So, choosing the particular case when W is $\mathbf{0}$ we have $n[\ ] \stackrel{\mathtt{pop}\langle n,h\rangle}{\longmapsto} \boldsymbol{\nu}\tilde{q}(n[\ _1] \mid\ _2)$ and $\boldsymbol{\nu}\tilde{p}(n[P_1] \mid P_2) \equiv K_1 \bullet \mathbf{0} \ \mathcal{S}\ K_2 \bullet \mathbf{0} \equiv \boldsymbol{\nu}\tilde{q}(n[\ _1] \mid\ _2)$, as desired.

– The inductive cases, for both kinds of actions, are straightforward.

• The remaining case, when $\boldsymbol{\nu}nP \ \mathcal{S}\ \boldsymbol{\nu}n$ because $P \ \mathcal{S}\ $, 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. . *Given any processes P and we have*

$$P \oplus \ \stackrel{\mathrm{d\ f}}{=}\ \boldsymbol{\nu}r(\mathtt{open}\langle r\rangle.P \mid \mathtt{open}\langle r\rangle. \ \mid r[\overline{\mathtt{open}}\langle r\rangle])$$

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

Note that up to structural congruence $P \oplus \ \stackrel{\tau}{\longrightarrow} P \mid \boldsymbol{\nu}r(\mathtt{open}\langle r\rangle. \ )$ and $P \oplus \ \stackrel{\tau}{\longrightarrow}\ \mid \boldsymbol{\nu}r(\mathtt{open}\langle r\rangle.P)$. 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 $P \oplus \ \stackrel{\tau}{\longrightarrow} P$ and $P \oplus \ \stackrel{\tau}{\longrightarrow} P$, 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* $P \stackrel{\mathtt{exit}\langle n,h\rangle}{\longmapsto} K$
- $P \stackrel{\alpha}{\longrightarrow} P'$ *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* $P \;\; \mathcal{S} \;\;$ *and* $P \stackrel{\alpha}{\longrightarrow}$

In order to emphasize the differences with the proof for delay bisimilarity we focus on one case, when $P \mid R \;\; \mathcal{S} \;\; \mid R$ because $P \;\; \mathcal{S} \;\;$. 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 $P \mid R \vdash\!\!\!\rightarrow^{\tau} O$.

Let us consider the case when $P \mid R \vdash\!\!\!\rightarrow^{\tau} O$ 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

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,* $'\mathrm{Y}_{\mathrm{b}}\langle$ *$SPdmTTdhTRTfTdTRTfTd,TTdTdπTRTfRTdandTRTTTdTRT$*

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

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

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

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

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 $\xRightarrow{\mathtt{enter}\langle n,h\rangle R}$ $'$ with P' $\mathcal{S}$ $'$. We define:

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

with a, h_i fresh. As $P \cong_{\mathrm{pop}}$ it

Examining the above reductions sequence from $C_{\alpha R}[\ ]$ we know that $\Rightarrow \xrightarrow{\mathtt{enter}\langle n,h\rangle R} C'[R]$. 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 $\stackrel{\mathtt{enter}\langle n,h\rangle R}{\Longrightarrow} C_2[R]$.

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 $\stackrel{\mathtt{exit}\langle n,h\rangle R}{\Longrightarrow}$ $'$ with P' $\mathcal{S}$ $'$. The proof strategy is the same as in the first case except that here we use the context

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

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

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

then there is a process Z such that

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

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 $C_{\alpha R}[\ ] \Rightarrow Z$ the prefix $\overline{\mathtt{o\ t}}\langle n,h\rangle$ 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 $'$ is $C_2[R]$; analysing the above reduction and applying Lemma 4.15(1) we obtain $\stackrel{\mathtt{exit}\langle n,h\rangle R}{\Longrightarrow}$ $'$, as required.

3. Let P $\overline{\mathtt{enter}}\langle n,h\rangle m[R]$

time we use the context

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

with h_i fresh. Arguing as usual from $C_{\alpha m[R]}[P] \cong_{\mathrm{pop}} C_{\alpha m[R]}[\ ]$, 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

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

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 $C_{\alpha m[R]}[\ ] \Rightarrow Z$ 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:

$$\begin{aligned} C_{\alpha m[R]}[\ ] &= \quad\quad \mid m[\mathtt{in}\langle n,h\rangle.(\mathrm{SPY_b}\langle n,h_1,h_2,R\rangle \oplus \mathtt{o\ t}\langle n,h_3\rangle)] \\ &\Longrightarrow \xrightarrow{\tau} C'[m[\mathrm{SPY_b}\langle n,h_1,h_2,R\rangle \oplus \mathtt{o\ t}\langle n,h_3\rangle]] \\ &\Longrightarrow \xrightarrow{\tau} C''[m[\mathrm{SPY_b}\langle n,h_1,h_2,R\rangle]] \\ &\Longrightarrow \quad C_2[m[\mathrm{SPY_b}\langle n,h_1,h_2,R\rangle]] \\ &= \quad Z \end{aligned}$$

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

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

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 $f[\boldsymbol{\nu} z(z[\mathtt{o\ t}\langle f,h\rangle])] \oplus f[\boldsymbol{\nu} z(z[\mathtt{o\ t}\langle f,h'\rangle])]$ 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 $'$ such that $\xRightarrow{\mathtt{in}\langle n,h\rangle}\ '$ and $P'\ \mathcal{S}\ '$. We define:

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

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

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

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

$Z \Downarrow_{\mathtt{pop}\langle f,h_2\rangle}$ and $Z \not\Downarrow_{\mathtt{pop}\langle f,h_3\rangle}$. This implies that in the reductions sequence $C_\alpha[\ ] \Rightarrow Z$ 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 $_1$, $_2$, $_3$, $'$ such that:

$$\begin{array}{rl}
C_\alpha[\ ] = & a[\ \ \mid \mathtt{o}\ \mathtt{t}\langle n,h_1\rangle.\overline{\mathtt{open}}\langle a\rangle] \mid n[\overline{\mathtt{in}}\langle n,h\rangle] \mid \\
& \overline{\mathtt{o}\ \mathtt{t}}\langle n,h_1\rangle.\mathtt{open}\langle a\rangle.(h_2 \oplus h_3) \\
\Longrightarrow \xrightarrow{\tau} & n[a[\ \ _1 \mid \mathtt{o}\ \mathtt{t}\langle n,h_1\rangle.\overline{\mathtt{open}}\langle a\rangle]] \mid \\
& \overline{\mathtt{o}\ \mathtt{t}}\langle n,h_1\rangle.\mathtt{open}\langle a\rangle.(h_2 \oplus h_3) \\
\Longrightarrow \xrightarrow{\tau} & a[\ \ _2 \mid \overline{\mathtt{open}}\langle a\rangle] \mid n[\,] \mid \mathtt{open}\langle a\rangle.(h_2 \oplus h_3) \\
\Longrightarrow \xrightarrow{\tau} & \ \ _3 \mid n[\,] \mid (h_2 \oplus h_3) \\
\Longrightarrow & \ \ ' \mid n[\,] \mid h_2 \\
= & Z \\
\cong_{\mathrm{pop}} & \ \ ' \mid h_2
\end{array}$$

where $\Rightarrow \xrightarrow{\mathtt{in}\langle n,h\rangle} {}_1 \Rightarrow {}_2 \Rightarrow {}_3 \Rightarrow {}'$. By Lemma 4.16(3) we can conclude that $P' \cong_{\mathrm{pop}} {}'$, as desired.

- Let $P \xrightarrow{\overline{\mathtt{in}}\langle n,h\rangle} P'$. Again we need to find some $'$ such that $\overline{\mathtt{in}}\langle n,h\rangle$

there exist $_1$, $_2$, and $_3$ such that:

$$
\begin{aligned}
C_\alpha[\ ] = \quad & n[\ \ | \ \overline{\texttt{open}}\langle n, h_1\rangle.(h_2 \oplus h_3)] \mid a[\texttt{in}\langle n, h\rangle.\texttt{o t}\langle n, h_4\rangle] \mid \\
& \overline{\texttt{o t}}\langle n, h_4\rangle.\texttt{open}\langle n, h_1\rangle \\
\Longrightarrow \xrightarrow{\tau} \ & n[\ \ _1 \mid \overline{\texttt{open}}\langle n, h_1\rangle.(h_2 \oplus h_3) \mid a[\texttt{o t}\langle n, h_4\rangle]] \mid \\
& \overline{\texttt{o t}}\langle n, h_4\rangle.\texttt{open}\langle n, h_1\rangle \\
\Longrightarrow \xrightarrow{\tau} \ & n[\ \ _2 \mid \overline{\texttt{open}}\langle n, h_1\rangle.(h_2 \oplus h_3)] \mid a[\ ] \mid \texttt{open}\langle n, h_1\rangle \\
\Longrightarrow \xrightarrow{\tau} \ & \ _3 \mid
\end{aligned}
$$

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 $\overline{\texttt{o t}}$, 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 $\overline{\texttt{o t}}\langle n\rangle$ 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 $(x).$ which on receiving a message binds it to x in which then executes. The basic reduction rule therefore takes the form

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

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.
- $(x).$, representing input of a message to be bound to x in .

We now have variables in our language, with the construct $(x).$ 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
	$\mid$	$\mathtt{o\ t}\langle n, h\rangle$	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
	$\mid$	$\overline{\mathtt{o\ t}}\langle n, h\rangle$	allow exit
	$\mid$	$\overline{\mathtt{open}}\langle n, h\rangle$	allow open

Expressions:

E, F ::= x

TABLE 8 Labelled Transition System - Communication

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

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

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

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, we write $P \approx$ if for all substitutions σ, mappings from v

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

$$\mathcal{S} = \{(P, \quad) : P \cong_{\mathrm{pop}} \quad , \quad P, \quad \text{processes}\}$$

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 $\quad'$ such that $\xRightarrow{(M)} \quad'$ and $P' \ \mathcal{S} \quad'$. As a distinguishing context take:

$$C_\alpha[\cdot] \stackrel{\mathrm{d\,f}}{=} [\cdot] \mid \langle M \rangle.(h_1 \oplus h_2)$$

  with h_1 and h_2 fresh.

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

$$C_{\alpha R}[\cdot] \stackrel{\mathrm{d\,f}}{=} [\cdot] \mid (x).(\mathrm{SPY}_{\mathrm{a}}\langle a, h_1, h_2, R\rangle \oplus a[b[\mathtt{o}\ \mathtt{t}\langle a, h_3\rangle]])$$

  with a, b, h_i fresh.

So we can conclude that for processes $P \cong \quad$ implies $P \approx \quad$.
Now consider two arbitrary terms $P, \quad$ such that $P \cong \quad$. We need to show that $P\sigma \approx \quad \sigma$ for any substitution σ. Let $x_1, x_2, \ldots, x_n$ be all the variables free in both P and $\quad$. From $P \cong \quad$ we know that

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

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

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

˜

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:

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

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

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.

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

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

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 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 carried by the agent is finally liberated inside the firewall by the opening of ambient a. Here is the protocol:

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

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

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

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

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. *$f\, h_a \notin \mathrm{fn}(P)$ and $h_f \notin \mathrm{fn}(\ \ )$, then:*

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

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

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 4.17, 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 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

[4] Luca Cardelli and Andrew D. Gordon. Types for mobile ambients. In *26th Annual S mposium 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 S mposium 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 e tended abstract appeared in *Proceedings of FoSSaCS ' 8*: 140–155.

[7] M. Hennessy and J. athke. Typed behavioural equivalences for processes in the presence of subtyping. Computer Science eport 2/01, University of Susse , 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] . Milner. *Communication and Concurrenc* . Prentice Hall, 1989.

[11] . Milner. The polyadic π-calculus: a tutorial. Technical eport ECS–LFCS–91–180, LFCS, Dept. of Comp.

[illegible]cto66.12030Tr90Td((2000.)Tj-200.039802Td(mobile)Also(D.)T