A Bisimulation-based Semantic Theory of Safe Ambients

MASSIMO MERRO
Dipartimento di Informatica, Università di Verona
and
MATTHEW HENNESSY
Department of Informatics, University of Sussex

We develop a semantics theory for SAP, a variant of Levi and Sangiorgi's Safe Ambients, SA.

The dynamics of SA relies upon *capabilities* (and *co-capabilities*) exercised by *mobile agents*, called *ambients*, to interact with each other. These capabilities contain references, the names of ambients with which they wish to interact. In SAP we generalise the notion of capability: in order to interact with an ambient n, an ambient m must exercise a capability indicating both n and a password h to access n; the interaction between n and m takes place only if n is willing to perform a corresponding co-capability with the same password h. The name h can also be looked upon as a *port* to *access ambient n via port h*.

In SAP by managing passwords/ports, 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/resources depending on the port accessed by the incoming clients. Then, we give an *lts*-based operational semantics for SAP and a labelled *bisimulation* equivalence which is proved to coincide with *reduction barbed congruence*.

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

Categories and Subject Descriptors: D.3.1 [**Programming languages**]: Formal Definition and Theory—*Synax*; *Semantics*; D.1.3 [**Programming Techniques**]: Concurrent Programming—*Distributed programming*; F.3.2 [**Meanings of Programs**]: Semantics of Programming Languages—*Operational semantics*

General Terms: Languages, Theory

Additional Key Words and Phrases: Mobile agents, distributed systems, bisimulation

1. INTRODUCTION

The calculus of *Mobile Ambients*, abbreviated MA, has been introduced by Cardelli and Gordon [2000] as a novel process calculus for describing *mobile agents*. The term

$$n[P]$$

Research funded by EPSRC grant GR/M71169.

An extended abstract appeared in the Conference Record of the *29th Symposium on Principle of Programming Languages*, pages 71-80, 2002.

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 π-calculus, [Milner et al. 1992], using the construct $\boldsymbol{\nu} nP$; here knowledge of n is restricted to P. For example the system

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

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

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

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

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

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

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

Papers such as [Cardelli and Gordon 2000; 1999] 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. This paper aims to provide a semantics theory for a significant variant of MA.

Before developing any algebraic theory for a process calculus at least two questions arise:

—What is the appropriate notion of semantic equivalence $\approx$ for that calculus?

—What proof methods exist for establishing such equivalences?

Bisimulation relations, in their various forms, have proved to be very popular as a basis for semantic equivalences for a variety of process calculi, such as CCS, [Milner 1989], and the π-calculus, [Milner et al. 1992]. Essentially the behaviour of processes is characterised using co-inductive relations defined over a *labelled transition system*, or *lts*, a collection of relations of the form

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

Intuitively this means that the system P may perform the action α, typically by interacting with its environment or context, and be thereby transformed into the system Q. The co-inductive nature of these equivalences ensures that there are powerful proof techniques available for establishing identities, [Sangiorgi 1992; Sangiorgi and Milner 1992; Sangiorgi 1994], addressing the second question above. Arguing for their appropriateness, the first question, is usually carried out by *contextual reasoning*. Intuitively the actions α in the judgement $P \xrightarrow{\alpha} Q$ represent some small context with which P can interact; more importantly it is shown that this collection of small contexts, codified as actions, are sufficient to capture all possible interactions which processes can have with arbitrary contexts. In short the bisimulation relation over the lts characterises some naturally defined contextually defined behavioural equivalence, [Sangiorgi 1992; Amadio et al. 1998]. This is the topic of the current paper:

> *Can we define an lts based operational semantics for an ambient-like calculus, and an associated bisimulation equivalence, which can be justified contextually?*

Levi and Sangiorgi [2000] argue that the calculus MA, as given in, for example [Cardelli and Gordon 2000], is qualitatively different from more standard process calculi such as the π-calculus. It is difficult for ambients to control potential interferences from other ambients. For example ambients are always under the threat of being entered by an arbitrary ambient located in the 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 Levi and Sangiorgi 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 π-calculus. 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

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

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*, has a much more satisfactory equational theory, and numerous equations, often type dependent, may be found in [Levi and Sangiorgi 2000]. 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.

In the current paper 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, $\overline{\texttt{in}}\langle n, h\rangle$. Actually, passwords can be looked upon as *ports*; for instance, the capability $\texttt{in}\langle n, h\rangle$ can be read as "*access ambient n via port h*". By managing passwords/ports, 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/resources depending on the passwords (respectively, ports) exhibited (respectively, accessed) by its clients.

Notice that the $\overline{\texttt{in}}\langle n, h\rangle$ co-capability represents an entry point for mobile ambients coming from *outside n*. This co-capability is exercised by the target computation space, i.e. ambient n. In our calculus, unlike Levi and Sangiorgi's SA, the co-capability $\overline{\texttt{out}}\langle n, h\rangle$ has a similar semantics: it models an entry point for mobile ambients coming from *inside n*. As a consequence, the co-capability $\overline{\texttt{out}}\langle n, h\rangle$ is exercised by the target computation space of the out-move, which is not n, but the current parent of n.

We call our calculus *Safe Ambients with Passwords*, abbreviated SAP. It is formally defined, with a reduction semantics in Section 2. It should be clear that SAP *is basically a generalisation/extension of* SA, as capabilities of the form $\texttt{in}\langle n\rangle$ can be seen as shorthand for $\texttt{in}\langle n, n\rangle$, where the name of an ambient is used as a password to access the ambient itself. However, the two main differences between SAP and SA, i.e. passwords and the semantics of $\overline{\texttt{out}}$, turn out to be crucial in the development of the semantics theory, and will be carefully discussed in the Conclusion.

Following the ideas of [Honda and Yoshida 1995; Milner and Sangiorgi 1992] it is straightforward to define a contextual equivalence between terms in SAP, or indeed any of the many other variants of ambients. We call *reduction barbed congruence*, $\cong$, the largest equivalence relation between terms which

—is a congruence for the language, that is is preserved by all constructs of the language

—preserves, in some sense, the reduction semantics of the language

—preserves *barbs*, that is preserves some simple observational property of terms.

A formal definition of reduction barbed congruence is given in Definition 2.5; the only real parameter here is in the precise definition of the allowed *barbs*. As we shall see, in our setting the resulting equivalence is invariant with respect to a wide variety of possible barbs. This emphasises our opinion that the resulting equivalence $\cong$ is a reasonable semantic equivalence for SAP. It has all of the extensional properties we require of such a relation although it is very difficult to reason about; see for example the proof of the equational laws in [Levi and Sangiorgi 2000]. However bisimulation relations, because of their co-inductive nature, provide powerful proof techniques for establishing equivalences, [Sangiorgi 1992; Sangiorgi and Milner 1992; Sangiorgi 1994].

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 $\cong$

—a set of algebraic laws proved using our bisimilarity.

The lts, on which the operational semantics is based, contains actions for all of the capabilities and co-capabilities in the language (here, as in much of the paper, we will ignore passwords unless they play a central role in the discussion). The most simple actions are those induced by the capabilities, as for instance in

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

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 $\mathtt{in}\langle n\rangle$. When such an ambient movement happens we must prescribe

—which ambient enters n

—what residual code remains behind.

For example, in the system

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

ambient $m[P]$ may migrate into an ambient n, and Q is the residual code; in general the migrating ambient and the residual code may share private names.

We represent the possibility to exercise a capability via *pre-actions*, whereas (visible) *actions* model interactions with the environment. In order to understand the difference between pre-actions and actions, consider the system

$$\boldsymbol{\nu}\tilde{r}\big(m[\ \mathtt{in}\langle n\rangle.P\ ] \mid Q\big)\ \big|\ n[\overline{\mathtt{in}}\langle n\rangle.R]\ .$$

Here ambient m may choose either to enter the ambient n, giving rise to a τ-action, or to enter another ambient n, provided by the environment. We model these two possibilities using the following pre-action:

$$\boldsymbol{\nu}\tilde{r}\big(m[\ \mathtt{in}\langle n\rangle.P\ ] \mid Q\big) \xrightarrow{\mathtt{pre\_enter}\langle n\rangle} \boldsymbol{\nu}\tilde{r}\langle m[P]\rangle Q\ .$$

where $\boldsymbol{\nu}\tilde{r}\langle m[P]\rangle Q$ is a *concretion*, [Milner 1991; Sangiorgi 1996; Levi and Sangiorgi 2000]. Here $m[P]$ is the migrating ambient, n the target, Q the residual code, and $\tilde{r}$, with $n \notin \tilde{r}$, the shared names. This pre-action may interact with the sibling ambient n (which may perform a $\overline{\mathtt{pre\_enter}}\langle n\rangle$ action) giving rise to the following τ-action

$$\boldsymbol{\nu}\tilde{r}\big(m[\ \mathtt{in}\langle n\rangle.P\ ] \mid Q\big)\ \big|\ n[\overline{\mathtt{in}}\langle n\rangle.R]\ \xrightarrow{\tau}\ \boldsymbol{\nu}\tilde{r}\big(Q \mid n[\ m[P] \mid R]\big)\ .$$

Alternatively, ambient m may decide to enter an ambient n provided by the environment. In this case, we will use the above $\mathtt{pre\_enter}\langle n\rangle$ action to derive the following higher-order $\mathtt{enter}\langle n\rangle$ action

$$\boldsymbol{\nu}\tilde{r}\big(m[\ \mathtt{in}\langle n\rangle.P\ ] \mid Q\big) \mid n[\overline{\mathtt{in}}\langle n\rangle.R]\ \xrightarrow{\overline{\mathtt{enter}}\langle n\rangle}\ \boldsymbol{\nu}\tilde{r}\big(n[\ \circ \mid m[P]\ ] \mid Q\big)\ \mid\ n[\overline{\mathtt{in}}\langle n\rangle.R]$$

where $\circ$ is a placeholder, added to the calculus, to remind us that the ambient n provided by the environment must be instantiated, at "bisimulation-time", with a process (also provided by the environment). As the instantiation of the derivatives is postponed, the lts is in a *late* style. The details are given in Section 3.

As we are interested in weak bisimilarities in Section 4 we give a notion of *weak action.* This is easy to achieve as the placeholder $\circ$ in a derivative does not perform any action. The weak moves $\overset{\alpha}{\Longrightarrow}$ are defined in the standard manner as $\xrightarrow{\tau}^{*} \xrightarrow{\alpha} \xrightarrow{\tau}^{*}$. We use our lts to define a form of (weak) *bisimilarity*, denoted by $\approx$, where the derivatives, when containing a placeholder, are instantiated with an arbitrary process before being tested again. In addition to being higher-order, the resulting bisimilarity is in *early* style as the universal quantification over the processes provided by the environment precedes the existential one on the derivatives.

The main result, Theorem 4.11, shows that the bisimilarity coincides, and therefore completely characterises, reduction barbed congruence in SAP.

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 [Cardelli and Gordon 2000; Levi and Sangiorgi 2000]. In Section 6 we apply our bisimilarity to prove a collection of useful *algebraic laws.* We want to stress that *those laws have very simple proofs.* Essentially, it is sufficient to show that the relation composed of the single pair of the processes under consideration is a bisimulation. By no means the proofs based on contextual reasoning developed in [Levi and Sangiorgi 2000; Gordon and Cardelli 2002], are that simple. In the same section we give some examples and prove the correctness of the protocol, introduced in [Cardelli and Gordon 2000], for controlling access through a firewall. The paper ends with Section 7, containing a discussion of our results, and a comparison with related work.

2. THE CALCULUS SAP

In Table I we give the syntax of processes. This is basically the same as that in [Cardelli and Gordon 2000], except that each of the original capabilities has a co-capability, as in [Levi and Sangiorgi 2000], and that now each capability has an extra argument h denoting a *password.*

The constructs for inactivity, parallel composition, restriction and replicated prefixing are inherited from mainstream concurrent calculi, most notably the π-calculus [Milner et al. 1992]. The inactive process, $\mathbf{0}$, does nothing. Parallel composition is denoted by a binary operator, $P \mid Q$, that is commutative and associative. The restriction operator, $\boldsymbol{\nu} nP$, creates a new (unique) name n within a scope P. We have replicated prefixing, $!C.P$, (rather than full replication) to create as many parallel replicas as needed. As in the π-calculus replicated prefixing allows us to derive a simpler labelled transition system; however, the theory and results in this paper could be easily adapted for a calculus with full replication. We also recall that in the π-calculus (i) replicated input has the same expressive power as full replication [Honda and Yoshida 1994] and recursion [Milner 1991; Sangiorgi and Walker 2001a]; (ii) replicated input has a simpler semantics and is convenient for implementations.

Names:	$n, h, \ldots \in \mathbf{N}$	
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
Processes:		
P, Q, R ::=	$\mathbf{0}$	nil process
$\mid$	$P_1 \mid P_2$	parallel composition
$\mid$	$\boldsymbol{\nu} n P$	restriction
$\mid$	$C.P$	prefixing
$\mid$	$n[P]$	ambient
$\mid$	$!C.P$	replication

Table I. The Calculus SAP

Specific to the ambient calculus are the *ambient* construct, $n[P]$, and the *prefix* via capabilities, $C.P$. In $n[P]$, n is the name of the ambient and P is the process running inside the ambient. The process $C.P$ executes an action regulated by the capability C, and then continues as the process P. Capabilities are obtained from names; given a name n, the capability $\mathtt{in}\langle n, h\rangle$ allows entry into n with password h, the capability $\mathtt{out}\langle n, h\rangle$ allows exit out of n with password h, and the capability $\mathtt{open}\langle n, h\rangle$ allows the destruction of the boundary of ambient n using the password h. For the sake of simplicity, at this stage, we omit *communication*; it will be added in Section 5.

We use a number of notational conventions. Parallel composition has the lowest precedence among the operators. $\prod_{i \in I} P_i$ means the parallel composition of all processes P_i, for $i \in I$. $\tilde{n}$ denotes a tuple $n_1, \ldots, n_k$ of names. The process $C.C'.P$ is read as $C.(C'.P)$. We omit trailing dead processes, writing C for $C.\mathbf{0}$, and $n[\;]$ for $n[\mathbf{0}]$. 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 write $\boldsymbol{\nu}\tilde{n}P$ as an abbreviation for $\boldsymbol{\nu} n_1 \ldots \boldsymbol{\nu} n_k P$. 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.

A (monadic) *context* $C[\cdot]$ is a process with a hole inside, denoted by $[\cdot]$. A *static context* $S[\cdot]$ is a (monadic) context where the hole does not appear under prefix or replication. The contexts exhibited in the paper will always be monadic, unless otherwise specified.

$P \mid Q \equiv Q \mid P$	(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)
$!C.P \equiv C.P \mid !C.P$	(Struct Repl Par)
$\boldsymbol{\nu} n\boldsymbol{\nu} mP \equiv \boldsymbol{\nu} m\boldsymbol{\nu} nP$	(Struct Res Res)
$n \notin \mathrm{fn}(P)$ implies $\boldsymbol{\nu} n(P \mid Q) \equiv P \mid \boldsymbol{\nu} nQ$	(Struct Res Par)
$n \neq m$ implies $\boldsymbol{\nu} n(m[P]) \equiv m[\boldsymbol{\nu} nP]$	(Struct Res Amb)

Table II. Structural Congruence

Reduction Semantics

The dynamics of the calculus is given in the form of a reduction relation. As customary in process calculi, the reduction semantics is based on an auxiliary relation, called *structural congruence*, which brings the participants of a potential interaction to contiguous positions. Let us formalise these two concepts.

Definition 2.1. *A relation $\mathcal{R}$ over processes is said to be* contextual*, if it is preserved by all the operators in the language. Formally this means it must satisfy the rules:*

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

A relation $\mathcal{R}$ is said to be 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 p-contextual equivalence relation which satisfies the axioms and rules in Table II.

The *reduction semantics* is given in terms of a binary relation over processes, $P \to Q$, which intuitively means that P can evolve to Q in one computation step. It is defined to be the least p-contextual relation which satisfies the axioms and rules in Table III.

The axiom (Red In) describes how an ambient n may *migrate into* an ambient m. It must exercise the capability $\mathtt{in}\langle m, h\rangle$ for some password h, and at the same time m, the target computation space, must be willing to allow *immigration*, exercising the co-capability, $\overline{\mathtt{in}}\langle m, h\rangle$; note that the password must be the same.

Emigration from an ambient is described in the rule (Red Out), and is similar. The ambient n may attempt to emigrate from ambient m by exercising the capability $\mathtt{out}\langle m, h\rangle$; but the target computation space must allow entry, by exercising the corresponding co-capability with the same password, $\overline{\mathtt{out}}\langle m, h\rangle$.

Remark 2.2. *Note that in [Levi and Sangiorgi 2000] this co-capability is exercised by m rather than the target computation space; we feel that with our definition*

$$n[\mathtt{in}\langle m,h\rangle.P \mid Q] \;\big|\; m[\overline{\mathtt{in}}\langle m,h\rangle.R \mid S] \;\rightarrow\; m[n[P \mid Q] \mid R \mid S] \qquad \text{(Red In)}$$

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

$$\mathtt{open}\langle n,h\rangle.P \;\big|\; n[\,\overline{\mathtt{open}}\langle n,h\rangle.Q \mid R\,] \;\rightarrow\; P \;\mid\; Q \;\mid\; R \qquad \text{(Red Open)}$$

$$P \equiv Q \quad Q \rightarrow R \quad R \equiv S \;\text{ implies }\; P \rightarrow S \qquad \text{(Red Str)}$$

Table III. Reduction Rules

there is a clearer distinction between the role of an ambient in a reduction and the corresponding role of its environment.

Finally the axiom (Red Open) describes the circumstances under which the ambient n can be opened unleashing its content. Again it requires the co-operation of n, exercising the co-capability $\overline{\mathtt{open}}\langle n,h\rangle$, for some password h, before the capability $\mathtt{open}\langle n,h\rangle$ has an effect. The single rule (Red Str) merely states that reductions are made modulo structural congruence which is used to bring the participants of a potential interaction into contiguous positions.

In the sequel we will use $\rightarrow^*$ to denote the reflexive and transitive closure of $\rightarrow$.

Behavioural semantics

We end this section with a definition of what we believe to be an appropriate notion of behavioural equivalence in ambient calculi, based on a notion of observation: *reduction barbed congruence*, [Honda and Yoshida 1995; Sangiorgi and Walker 2001b]. Following [Honda and Yoshida 1995; Sangiorgi and Walker 2001b] we define reduction barbed congruence, $\cong$, as the *largest* symmetric relation over processes which satisfies the following criteria:

—it is *contextual*, to aid in compositional verification,

—it is invariant, in some sense, with respect to the reduction relation $\rightarrow$; formally, we say a relation $\mathcal{R}$ is *reduction closed* if $P\,\mathcal{R}\,Q$ and $P \rightarrow P'$ implies the existence of some Q' such that $Q \Rightarrow Q'$ and $P'\,\mathcal{R}\,Q'$;

—it preserves some intuitive *observation predicate*, $P{\downarrow_n}$.

When dealing with ambients there are many ways of formulating observation predicates. In [Cardelli and Gordon 2000] the predicate $P \downarrow_n$ is used to denote the possibility of process P of interacting with the environment via the ambient n; it is true whenever $P \equiv \boldsymbol{\nu}\tilde{m}(n[P_1] \mid P_2)$, where $n \notin \{\tilde{m}\}$. This is a reasonable definition of observation for MA as no authorisation is required to cross a boundary. As a consequence, the presence of an ambient n at top level denotes a potential interaction between the process and the environment via n. However in SA 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 SA, $P{\downarrow_n}$ is defined to be true whenever $P \equiv \boldsymbol{\nu}\tilde{m}(n[C.P_1 \mid P_2] \mid P_3)$ where $C \in \{\overline{\mathtt{in}}\langle n\rangle, \overline{\mathtt{open}}\langle n\rangle\}$ and $n \notin \{\tilde{m}\}$. We use a slight simplification of this definition.

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

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

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

Our choice of observation here may seem arbitrary. However, Theorem 3.7 shows that our contextual equality remains invariant under a large choice of possible observation predicates. Note also that our barbs only mention 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 will see (again in Theorem 3.7) this is unnecessary.

Definition 2.4. *A relation* $\mathcal{R}$ *over processes is said to be* barb preserving *if* $P \mathrel{\mathcal{R}} Q$ *and* $P{\downarrow_n}$ *implies* $Q{\Downarrow_n}$.

Definition Reduction barbed congruence. *Reduction barbed congruence, written* $\cong$, *is the largest symmetric relation over processes which is*

—*contextual*

—*reduction closed*

—*barb preserving.*

It is easy to prove that reduction barbed congruence is an *equivalence relation.*

The aim of the paper is to give a co-inductive characterisation of $\cong$ in SAP using an lts-based operational semantics.

3. A LABELLED TRANSITION SEMANTICS

In Tables V, VI, and VII, we propose a *labelled transition system*, lts, for a slight extension of our calculus. This extension allows us (i) to define the lts in a *late* style, in which the instantiation of the derivatives is postponed, and (ii) to adopt the standard definition for weak actions. The lts is obviously higher-order as it models agent mobility.

The capabilities or prefixes C in our language give rise, in the standard manner, [Milner 1989], to actions of the form $P \xrightarrow{C} Q$; for example we have

$$\overline{\texttt{in}}\langle n, h\rangle.P_1 \mid P_2 \xrightarrow{\overline{\texttt{in}}\langle n,h\rangle} P_1 \mid P_2.$$

These actions could be used to define a version of weak bisimulation equivalence over processes, $\approx_{\text{bad}}$, again in the standard manner, [Milner 1989]. However it should be obvious that $\approx_{\text{bad}}$ is unsatisfactory as a notion of equivalence for SAP. For example these actions cannot be performed by ambients and therefore we would have the wrong identity

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

regardless of P.

Nevertheless, the capabilities above can be considered the basis of further actions. For example, the system

$$n[\texttt{in}\langle m\rangle.P] \mid Q$$

has the capability to *enter* ambient m. Exercising this capability has a dual effect; on the one hand the ambient $n[P]$ will 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 prefixes C will induce more complicated actions in

Pre-Actions:	μ	$::=$	$\mathtt{pre\_enter}\langle n,h\rangle \mid \mathtt{pre\_exit}\langle n,h\rangle$	
		$\mid$	$\overline{\mathtt{pre\_enter}}\langle n,h\rangle$	
Actions:	α	$::=$	τ	
		$\mid$	$\mathtt{in}\langle n,h\rangle \mid \mathtt{out}\langle n,h\rangle \mid \mathtt{open}\langle n,h\rangle$	
		$\mid$	$\overline{\mathtt{in}}\langle n,h\rangle \mid \overline{\mathtt{out}}\langle n,h\rangle \mid \overline{\mathtt{open}}\langle n,h\rangle$	
		$\mid$	$\mathtt{enter}\langle n,h\rangle \mid \mathtt{exit}\langle n,h\rangle$	
		$\mid$	$\overline{\mathtt{enter}}\langle n,h\rangle \mid \mathtt{free}\langle n,h\rangle$	
		$\mid$	$\mathtt{pop}\langle n,h\rangle$	
Labels:	λ	$::=$	$\mu \mid \alpha$	
Extended processes:	E, F	$::=$	$\circ$	placeholder
		$\mid$	$\mathbf{0}$	nil process
		$\mid$	$E_1 \mid E_2$	parallel composition
		$\mid$	$\boldsymbol{\nu} nE$	restriction
		$\mid$	$C.E$	prefixing
		$\mid$	$n[E]$	ambient
		$\mid$	$!C.E$	replication
Concretions:	K	$::=$	$\boldsymbol{\nu}\tilde{m}\langle E\rangle F$	
Outcomes:	O	$::=$	$E \mid K$	

Table IV. Actions, Extended processes, Concretions and Outcomes

ambients, and more generally in processes. These will be formulated as transitions of the form

$$E \xrightarrow{\lambda} O$$

where λ denotes a *label* and E and O range over *extended processes* and *outcomes*, respectively, as defined in Table IV.

Essentially, we extend the syntax of processes with a *special process* $\circ$ to pinpoint those ambients whose content will be instantiated later, with a process provided by the environment. The special process $\circ$ does not reduce and does not interact with anyone: it is simply a placeholder. The generalisation of structural congruence to extended processes is straightforward, supposing $\mathrm{fn}(\circ) = \emptyset$. We call *pure processes* those processes which do not contain $\circ$. Obviously, the lts also applies to pure processes.

We make a distinction between *pre-actions*, of the form $E \xrightarrow{\mu} K$, where K ranges over *concretions* (Table IV), and *actions*, of the form $E \xrightarrow{\alpha} E'$. Pre-actions denote the possibility to exercise a move capability while actions model the interaction of a process with its environment. As usual, we also have τ*-actions* to model internal computations. As pre-actions express the possibility to exercise a capability, only actions model the evolution of a process at run-time.

Pre-actions are generated via the rules (Pre-Enter), (Pre-Co-Enter), and (Pre-Exit) of Table V. Their general form is

$$E \xrightarrow{\mu} \boldsymbol{\nu}\tilde{m}\langle E_1\rangle E_2$$

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

$$\text{(Pre-Enter)}\ \frac{E \xrightarrow{\mathtt{in}\langle n,h\rangle} E'}{m[E] \xrightarrow{\mathtt{pre\_enter}\langle n,h\rangle} \langle m[E']\rangle\mathbf{0}} \ \text{(Pre-Co-Enter)}\ \frac{E \xrightarrow{\overline{\mathtt{in}}\langle n,h\rangle} E'}{n[E] \xrightarrow{\overline{\mathtt{pre\_enter}}\langle n,h\rangle} \langle E'\rangle\mathbf{0}}$$

$$(\tau\ \text{Enter})\ \frac{E \xrightarrow{\mathtt{pre\_enter}\langle n,h\rangle} \boldsymbol{\nu}\tilde{p}\langle E_1\rangle E_2 \qquad F \xrightarrow{\overline{\mathtt{pre\_enter}}\langle n,h\rangle} \boldsymbol{\nu}\tilde{q}\langle F_1\rangle F_2}{\begin{array}{c} E \mid F \xrightarrow{\tau} \boldsymbol{\nu}\tilde{p}\boldsymbol{\nu}\tilde{q}(n[E_1 \mid F_1] \mid E_2 \mid F_2) \\ F \mid E \xrightarrow{\tau} \boldsymbol{\nu}\tilde{p}\boldsymbol{\nu}\tilde{q}(n[F_1 \mid E_1] \mid F_2 \mid E_2) \end{array}}$$

$$\text{if}\ \ ((\text{fn}(E_1) \cup \text{fn}(E_2)) \cap \{\tilde{q}\}) = ((\text{fn}(F_1) \cup \text{fn}(F_2)) \cap \{\tilde{p}\}) = \emptyset$$

$$\text{(Pre-Exit)}\ \frac{E \xrightarrow{\mathtt{out}\langle n,h\rangle} E'}{m[E] \xrightarrow{\mathtt{pre\_exit}\langle n,h\rangle} \langle m[E']\rangle\mathbf{0}} \qquad \text{(Pop)}\ \frac{E \xrightarrow{\mathtt{pre\_exit}\langle n,h\rangle} \boldsymbol{\nu}\tilde{m}\langle E_1\rangle E_2}{n[E] \xrightarrow{\mathtt{pop}\langle n,h\rangle} \boldsymbol{\nu}\tilde{m}(n[E_2] \mid E_1)}$$

$$(\tau\ \text{Exit})\ \frac{E \xrightarrow{\mathtt{pop}\langle n,h\rangle} E' \quad F \xrightarrow{\overline{\mathtt{out}}\langle n,h\rangle} F'}{\begin{array}{c} E \mid F \xrightarrow{\tau} E' \mid F' \\ F \mid E \xrightarrow{\tau} F' \mid E' \end{array}}$$

$$\text{(Free)}\ \frac{E \xrightarrow{\overline{\mathtt{open}}\langle n,h\rangle} E'}{n[E] \xrightarrow{\mathtt{free}\langle n,h\rangle} E'} \qquad (\tau\ \text{Open})\ \frac{E \xrightarrow{\mathtt{open}\langle n,h\rangle} E' \quad F \xrightarrow{\mathtt{free}\langle n,h\rangle} F'}{\begin{array}{c} E \mid F \xrightarrow{\tau} E' \mid F' \\ F \mid E \xrightarrow{\tau} F' \mid E' \end{array}}$$

Table V. Labelled Transition System - Enter, Exit, and Open

where E_1 represents the code that may enter to, reside at, or exit from an ambient, while E_2 represents the derivative which is not affected by the move, and $\tilde{m}$ is the set of private names shared by E_1 and E_2. We adopt the convention that if K is the concretion $\boldsymbol{\nu}\tilde{m}\langle E\rangle F$, then $\boldsymbol{\nu}rK$ is a shorthand for $\boldsymbol{\nu}\tilde{m}\langle E\rangle\boldsymbol{\nu}rF$, if $r \notin \text{fn}(E)$, and the concretion $\boldsymbol{\nu}r\tilde{m}\langle E\rangle F$ otherwise. We have a similar convention for the rule (π Par): $K \mid E'$ is defined to be the concretion $\boldsymbol{\nu}\tilde{m}\langle E\rangle(F \mid E')$, where $\tilde{m}$ are chosen, using α-conversion if necessary, so that $\text{fn}(E') \cap \tilde{m} = \emptyset$; similarly $E' \mid K$ is the concretion $\boldsymbol{\nu}\tilde{m}\langle E\rangle(E' \mid F)$. Occasionally, we omit dead processes when they are in parallel with processes, writing E for $E \mid \mathbf{0}$.

Pre-actions are used to construct the internal actions for entering and exiting ambients of Table V. The rule (τ Enter) models an ambient moving into a sibling ambient. The rule (Pop) models an ambient exiting from another one. This action is not yet internal as it requires the presence of the corresponding $\overline{\mathtt{out}}$ co-capability, as described in rule (τ Exit). The rules for opening ambients are straightforward.

$$\text{(Enter)}\ \frac{E \xrightarrow{\texttt{pre\_enter}\langle n,h\rangle} \boldsymbol{\nu}\tilde{m}\langle E_1\rangle E_2}{E \xrightarrow{\texttt{enter}\langle n,h\rangle} \boldsymbol{\nu}\tilde{m}(n[\circ \mid E_1] \mid E_2)}$$

$$\text{(Co-Enter)}\ \frac{E \xrightarrow{\overline{\texttt{pre\_enter}}\langle n,h\rangle} \boldsymbol{\nu}\tilde{m}\langle E_1\rangle E_2}{E \xrightarrow{\overline{\texttt{enter}}\langle n,h\rangle} \boldsymbol{\nu}\tilde{m}(n[\circ \mid E_1] \mid E_2)} \qquad \text{(Exit)}\ \frac{E \xrightarrow{\texttt{pre\_exit}\langle n,h\rangle} \boldsymbol{\nu}\tilde{m}\langle E_1\rangle E_2}{E \xrightarrow{\texttt{exit}\langle n,h\rangle} \boldsymbol{\nu}\tilde{m}(n[\circ \mid E_2] \mid E_1)}$$

Table VI. Labelled Transition System - Higher-order rules for ambient mobility

$$(\tau\ \text{Amb})\ \frac{E \xrightarrow{\tau} E'}{n[E] \xrightarrow{\tau} n[E']}$$

$$\text{(Par)}\ \frac{E \xrightarrow{\lambda} O \qquad \lambda \notin \{\texttt{enter}\langle n,h\rangle, \texttt{exit}\langle n,h\rangle, \overline{\texttt{enter}}\langle n,h\rangle\}}{\begin{array}{c} E \mid F \xrightarrow{\lambda} O \mid F \\ F \mid E \xrightarrow{\lambda} F \mid O \end{array}}$$

$$\text{(Res)}\ \frac{E \xrightarrow{\lambda} O \qquad \lambda \notin \{\texttt{enter}\langle n,h\rangle, \texttt{exit}\langle n,h\rangle, \overline{\texttt{enter}}\langle n,h\rangle\} \quad n \notin \text{fn}(\lambda)}{\boldsymbol{\nu}nE \xrightarrow{\lambda} \boldsymbol{\nu}nO}$$

Table VII. Labelled Transition System - Structural Rules

The higher-order rules (Enter) and (Exit) of Table VI turn concretions into systems by explicitly introducing the ambient entered or exited by the process in question. The higher-order rule (Co-Enter) models an ambient n accepting an incoming ambient. All the derivatives of higher-order rules contain an ambient n with a placeholder inside. This placeholder reminds us that the content of n must be instantiated later, in the bisimilarity, with an arbitrary process provided by the environment.

The structural rules (τ Amb), (Par), and (Res) of Table VII are straightforward; just notice that higher-order actions $\texttt{enter}$, $\overline{\texttt{enter}}$, and $\texttt{exit}$ do not have structural rules as they model "final" higher-order interactions with the environment.

Now, let us explain our lts with an example. Let us first examine those rules induced by the prefix $\texttt{in}$, 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 follows:

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

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

$$m[\texttt{in}\langle n\rangle.P] \xrightarrow{\texttt{pre\_enter}\langle n\rangle} \langle m[P]\rangle\mathbf{0}\ .$$

More generally, using the structural rule (Par) we have:

$$P_2 \mid m[\mathtt{in}\langle n\rangle.P_1] \xrightarrow{\mathtt{pre\_enter}\langle n\rangle} \langle m[P_1]\rangle P_2 \ .$$

This means that the system $P_2 \mid m[\mathtt{in}\langle n\rangle.P_1]$ 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. The above pre-action can interact with an action $\overline{\mathtt{enter}}\langle n\rangle$ performed by n. The rule (Co-Enter) allows these to be derived. So for example, using again (Par), we have

$$Q_2 \mid n[\overline{\mathtt{in}}\langle n\rangle.Q_1] \xrightarrow{\overline{\mathtt{pre\_enter}}\langle n\rangle} \langle Q_1\rangle Q_2 \ .$$

Here, after the co-action, Q_1 remains inside n, while Q_2 is outside, and the placeholder $\circ$ models the entry point for the ambient moving into n. Now, the communication rule (τ Enter) 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:

$$\begin{aligned} P_2 \mid m[\mathtt{in}\langle n\rangle.P_1] \mid Q_2 \mid n[\overline{\mathtt{in}}\langle n\rangle.Q_1] &\xrightarrow{\tau} n[\,m[P_1] \mid Q_1] \mid P_2 \mid Q_2 \\ &\equiv \ P_2 \mid Q_2 \mid n[\,m[P_1] \mid Q_1] \end{aligned}$$

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

The structural rule (Res) allows the migrating ambient to share private names with its point of origin, in the same manner as in the π-calculus. So, for example if k occurs free in both P_1 and P_2 of the above τ-action, then we have the action

$$\boldsymbol{\nu} k(P_2 \mid m[\mathtt{in}\langle n\rangle.P_1]) \xrightarrow{\mathtt{pre\_enter}\langle n\rangle} \boldsymbol{\nu} k\langle m[P_1]\rangle P_2$$

and the rule (τ Enter) now gives

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

where it is assumed that k is chosen to be fresh to n, Q_1 and Q_2. Note that the scope of k has now extended to include the ambient n.

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

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

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

$$m[\,\mathtt{out}\langle n\rangle.P_1\,] \mid P_2 \xrightarrow{\mathtt{pre\_exit}\langle n\rangle} \langle m[P_1]\rangle P_2 \ .$$

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. If the system under consideration already contains the ambient n to exit from, then we have another action, called $\mathtt{pop}\langle n\rangle$, with the associated rule (Pop); an application of which gives:

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

The action $\mathtt{pop}\langle n\rangle$ can interact with the co-action $\overline{\mathtt{out}}\langle n\rangle$ as codified in the rule (τ Exit); an application of which gives the original move above:

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

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

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

and an application of the rule (τ Open) gives

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

As in other concurrent calculi, for any transition $P \xrightarrow{\lambda} O$, the structure of P and O can be determined up to structural congruence.

Lemma 3.1.

(1) If $P \xrightarrow{C} P'$, *with* $C \in \{\mathtt{in}\langle n,h\rangle,\ \mathtt{out}\langle n,h\rangle,\ \mathtt{open}\langle n,h\rangle,\ \overline{\mathtt{in}}\langle n,h\rangle,\ \overline{\mathtt{out}}\langle n,h\rangle,\ \overline{\mathtt{open}}\langle n,h\rangle\}$, *then there exist* $\tilde{p}, P_1, P_2$, *with* $n,h \notin \tilde{p}$, *such that*

$$P \equiv \boldsymbol{\nu}\tilde{p}(C.P_1 \mid P_2) \quad \textit{and} \quad P' \equiv \boldsymbol{\nu}\tilde{p}(P_1 \mid P_2)$$

(2) if $P \xrightarrow{\mathtt{pre\_enter}\langle n,h\rangle} \boldsymbol{\nu}\tilde{p}\langle P'\rangle P''$ *then there exist* k, P_1, P_2, *with* $n,h \notin \tilde{p}$, *such that*

$$P \equiv \boldsymbol{\nu}\tilde{p}(k[\mathtt{in}\langle n,h\rangle.P_1 \mid P_2] \mid P'') \quad \textit{and} \quad P' \equiv k[P_1 \mid P_2]$$

(3) if $P \xrightarrow{\mathtt{pre\_exit}\langle n,h\rangle} \boldsymbol{\nu}\tilde{p}\langle P'\rangle P''$ *then there exist* k, P_1, P_2, *with* $n,h \notin \tilde{p}$, *such that*

$$P \equiv \boldsymbol{\nu}\tilde{p}(k[\mathtt{out}\langle n,h\rangle.P_1 \mid P_2] \mid P'') \quad \textit{and} \quad P' \equiv k[P_1 \mid P_2]$$

(4) if $P \xrightarrow{\overline{\mathtt{pre\_enter}}\langle n,h\rangle} \boldsymbol{\nu}\tilde{p}\langle P'\rangle P''$ *then there exist* P_1, P_2, *with* $n,h \notin \tilde{p}$, *such that*

$$P \equiv \boldsymbol{\nu}\tilde{p}(n[\overline{\mathtt{in}}\langle n,h\rangle.P_1 \mid P_2] \mid P'') \quad \textit{and} \quad P' \equiv P_1 \mid P_2$$

(5) if $P \xrightarrow{\mathtt{free}\langle n,h\rangle} P'$ *then there exist* P_1, P_2, P_3, *with* $n,h \notin \tilde{p}$, *such that*

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

(6) if $P \xrightarrow{\mathtt{pop}\langle n,h\rangle} P'$ *then there exist* k, P_1, P_2, P_3, P_4, *with* $n,h \notin \tilde{p}$, *such that*

$$P \equiv \boldsymbol{\nu}\tilde{p}(n[k[\mathtt{out}\langle n,h\rangle.P_1 \mid P_2] \mid P_3] \mid P_4) \quad \textit{and} \quad P' \equiv \boldsymbol{\nu}\tilde{p}(n[P_3] \mid k[P_1 \mid P_2] \mid P_4)\ .$$

Proof. By induction on the transition rules of Tables V, VI, and VII. □

By part 2 and 3 of Lemma 3.1 only ambients, rather than general code, can migrate.

We end this section showing that, on pure processes, the lts-based semantics coincides with the reduction semantics of Section 2.

Theorem 3.2.

(1) *If* $P \xrightarrow{\tau} P'$ *then* $P \to P'$.

(2) *If* $P \to P'$ *then* $P \xrightarrow{\tau}\equiv P'$.

Proof. By transition induction. Part 1 is the most difficult. We recall that τ-transitions can only be generated by the rules in Tables V and VII. Let's prove the most significant cases.

(τ Enter) In this case $P = P_1 \mid P_2$, and the silent move is due to the following visible actions: $P_1 \xrightarrow{\texttt{pre\_enter}\langle n,h\rangle} \boldsymbol{\nu}\tilde{p}\langle Q_1\rangle R_1$, $P_2 \xrightarrow{\overline{\texttt{pre\_enter}}\langle n,h\rangle} \boldsymbol{\nu}\tilde{q}\langle Q_2\rangle R_2$. Thus,

$$P_1 \mid P_2 \xrightarrow{\tau} \boldsymbol{\nu}\tilde{p}\boldsymbol{\nu}\tilde{q}(n[Q_1 \mid Q_2] \mid R_1 \mid R_2)\ .$$

By Lemma 3.1(2) we deduce that $P_1 \equiv \boldsymbol{\nu}\tilde{p}(k[\texttt{in}\langle n,h\rangle.P_1' \mid P_1''] \mid R_1)$, $Q_1 \equiv k[P_1' \mid P_1'']$, for some P_1' and P_1''. Lemma 3.1(4) guarantees that $P_2 \equiv \boldsymbol{\nu}\tilde{q}(n[\overline{\texttt{in}}\langle n,h\rangle.P_2' \mid P_2''] \mid R_2)$ and $Q_2 \equiv P_2' \mid P_2''$, for some P_2' and P_2''. Then,

$$\begin{aligned}
P_1 \mid P_2 &\equiv \boldsymbol{\nu}\tilde{p}(k[\texttt{in}\langle n,h\rangle.P_1' \mid P_1''] \mid R_1) \mid \boldsymbol{\nu}\tilde{q}(n[\overline{\texttt{in}}\langle n,h\rangle.P_2' \mid P_2''] \mid R_2)\\
&\equiv \boldsymbol{\nu}\tilde{p}\boldsymbol{\nu}\tilde{q}(k[\texttt{in}\langle n,h\rangle.P_1' \mid P_1''] \mid n[\overline{\texttt{in}}\langle n,h\rangle.P_2' \mid P_2''] \mid R_1 \mid R_2)\\
&\to \boldsymbol{\nu}\tilde{p}\boldsymbol{\nu}\tilde{q}(n[k[P_1' \mid P_1''] \mid P_2' \mid P_2''] \mid R_1 \mid R_2)\\
&\equiv \boldsymbol{\nu}\tilde{p}\boldsymbol{\nu}\tilde{q}(n[Q_1 \mid Q_2] \mid R_1 \mid R_2)\ .
\end{aligned}$$

By applying the rule (Red Str) we obtain $P \to \boldsymbol{\nu}\tilde{p}\boldsymbol{\nu}\tilde{q}(n[Q_1 \mid Q_2] \mid R_1 \mid R_2)$, as desired.

(τ Exit) In this case $P = P_1 \mid P_2$, $P_1 \xrightarrow{\texttt{pop}\langle n,h\rangle} Q_1$, $P_2 \xrightarrow{\overline{\texttt{out}}\langle n,h\rangle} Q_2$, and

$$P_1 \mid P_2 \xrightarrow{\tau} Q_1 \mid Q_2\ .$$

By Lemma 3.1(6) we deduce that $P_1 \equiv \boldsymbol{\nu}\tilde{p}(n[k[\texttt{out}\langle n,h\rangle.P_1' \mid P_1''] \mid P_1'''] \mid P_1'''')$ and $Q_1 \equiv \boldsymbol{\nu}\tilde{p}(n[P_1'''] \mid k[P_1' \mid P_1''] \mid P_1'''')$, for some $\tilde{p}$, k, P_1', P_1'', P_1''', P_1''''. Lemma 3.1(1) guarantees that $P_2 \equiv \boldsymbol{\nu}\tilde{q}(\overline{\texttt{out}}\langle n,h\rangle.P_2' \mid P_2'')$ and $Q_2 \equiv \boldsymbol{\nu}\tilde{q}(P_2' \mid P_2'')$, for some $\tilde{q}$, P_2', and P_2''. Then,

$$\begin{aligned}
P_1 \mid P_2 &\equiv \boldsymbol{\nu}\tilde{p}\boldsymbol{\nu}\tilde{q}(n[k[\texttt{out}\langle n,h\rangle.P_1' \mid P_1''] \mid P_1'''] \mid \overline{\texttt{out}}\langle n,h\rangle.P_2' \mid P_1'''' \mid P_2'')\\
&\to \boldsymbol{\nu}\tilde{p}\boldsymbol{\nu}\tilde{q}(n[P_1'''] \mid k[P_1' \mid P_1''] \mid P_2' \mid P_1'''' \mid P_2'')\\
&\equiv Q_1 \mid Q_2\ .
\end{aligned}$$

By applying the rule (Red Str) we obtain $P \to Q_1 \mid Q_2$, as desired. □

3.1 Barbs and actions

Here we re-examine our definition of reduction barbed congruence, $\cong$, showing that it is very robust under changes to the precise definition of barbs.

As already mentioned in Section 2, according to [Cardelli and Gordon 2000; Levi and Sangiorgi 2000], the predicate $P{\downarrow_n}$ detects the ability of a process P to interact with its environment via the ambient n. However, in other process calculi, like the π-calculus, barbs are defined using (visible) actions. So, one may wonder how our definition of reduction barbed congruence would be affected by inheriting the notion of barb from the lts. In fact we can show that our definition of barb coincides with the choice of a particular action:

Lemma 3.3. *$P{\downarrow_n}$ iff $P \xrightarrow{\texttt{free}\langle n,h\rangle} P'$ for some h and P'.*

Proof. Straightforward. □

Now we prove that for all possible actions (different from τ) the resulting definitions of reduction barbed congruence collapse and coincide with $\cong$. We recall that α ranges over the actions defined in Table IV.

Definition 3.4. *We write $P{\downarrow_\alpha}$ if $P \xrightarrow{\alpha}$. We write $P{\Downarrow_\alpha}$ if $P \xrightarrow{\tau}{}^* \xrightarrow{\alpha}$.*

Definition 3.5. *Let $\mathcal{A} = \{\texttt{in}, \texttt{out}, \texttt{open}, \overline{\texttt{in}}, \overline{\texttt{out}}, \overline{\texttt{open}}, \texttt{enter}, \overline{\texttt{enter}}, \texttt{exit}, \texttt{pop}, \texttt{free}\}$. For any $\rho \in \mathcal{A}$, let $\cong_\rho$ be the largest symmetric relation over pure processes which*

—is contextual

—is reduction closed

—if $P \cong_\rho Q$ and $P \downarrow_{\rho\langle n,h\rangle}$ then $Q \Downarrow_{\rho\langle n,h\rangle}$.

The next lemma is a well-known result for barbed bisimilarities which follows from Theorem 3.2.

Lemma 3.6. *If $P \cong_\rho Q$ then*

(1) $P{\Downarrow_n}$ iff $Q{\Downarrow_n}$

(2) $P \xrightarrow{\tau}{}^* P'$ implies $Q \xrightarrow{\tau}{}^* Q'$ for some Q' such that $P' \cong_\rho Q'$.

Now we are ready to prove that our definition of reduction barbed congruence remains invariant under changes of ρ-barb.

Theorem 3.7. *Let P and Q be two processes, then*

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

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

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

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

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

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

This is sufficient to establish the result. For $P \cong Q$ implies $S_1[P] \cong S_1[Q]$ which in turn implies $S_1[Q]\Downarrow_f$, from which we have the required $Q\Downarrow_{\texttt{pop}\langle n,h\rangle}$. As to the implication from right to left, let $P \cong_{\text{pop}} Q$ and $P\downarrow_n$, then we want to conclude that $Q\Downarrow_n$. Again this involves using a context. By Lemma 3.3 if $P\downarrow_n$ then there exists h such that $P \xrightarrow{\texttt{free}\langle n,h\rangle}$. Thus, we define a context:

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

This context is constructed so that whenever r and k are fresh to R then:
(a) $S_2^h[R]\Downarrow_{\texttt{pop}\langle k\rangle}$ implies $R\Downarrow_n$
(b) $R\Downarrow_n$ implies $\exists h.\ S_2^h[R]\Downarrow_{\texttt{pop}\langle k\rangle}$.
This is sufficient to establish $Q\Downarrow_n$.

(2) $\rho = \overline{\texttt{enter}}$.
Again this is a question of defining two appropriate contexts. Let

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

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

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

This context has the required property that:
(a) $S_2^h[R]\Downarrow_{\overline{\texttt{enter}}\langle g\rangle}$ implies $R\Downarrow_n$
(b) $R\Downarrow_n$ implies $\exists h.\ S_2^h[R]\Downarrow_{\overline{\texttt{enter}}\langle g\rangle}$.
provided that g is fresh to R.

□

REMARK 3.8. *Note that* the use of passwords is fundamental to the above result. *In particular, in the case* $\rho = \overline{\texttt{enter}}$ *the use of the fresh password* k *in the definition of* $S_1[\cdot]$ *is essential. Note also that this case,* $\rho = \overline{\texttt{enter}}$*, shows that Levi and Sangiorgi's definition of barb, [Levi and Sangiorgi 2000], can be simplified, to coincide with our original definition.*

4. THE CHARACTERISATION

Since we are interested in weak bisimilarities we have to provide a notion of *weak action*. In our setting, weak actions can be defined in the standard manner.

DEFINITION WEAK ACTIONS.

(1) $\overset{\alpha}{\Longrightarrow}$ *denotes* $\xrightarrow{\tau}{}^{*} \xrightarrow{\alpha} \xrightarrow{\tau}{}^{*}$

(2) $\overset{\hat{\alpha}}{\Longrightarrow}$ *denotes* $\xrightarrow{\tau}{}^{*}$ *if* $\alpha = \tau$ *and* $\overset{\alpha}{\Longrightarrow}$ *otherwise.*

Notice that the τ-moves following a visible action α may cause the placeholder $\circ$ to move into other ambients. For instance:

$$\begin{array}{rc} k[\mathtt{out}\langle n\rangle.\overline{\mathtt{in}}\langle k\rangle.\mathtt{open}\langle n\rangle.P] \mid \mathtt{in}\langle k\rangle.\overline{\mathtt{open}}\langle n\rangle.Q & \xrightarrow{\mathtt{exit}\langle n\rangle} \\ n[\circ \mid \mathtt{in}\langle k\rangle.\overline{\mathtt{open}}\langle n\rangle.Q] \mid k[\overline{\mathtt{in}}\langle k\rangle.\mathtt{open}\langle n\rangle.P] & \xrightarrow{\tau} \\ k[n[\circ \mid \overline{\mathtt{open}}\langle n\rangle.Q] \mid \mathtt{open}\langle n\rangle.P] & \xrightarrow{\tau} \\ k[\circ \mid Q \mid P] & \end{array}$$

where the placeholder has moved from ambient n into ambient k.

Notice also that the derivative E of a weak action $P \overset{\alpha}{\Longrightarrow} E$ may contain at most one placeholder $\circ$, although arbitrary nested.

Proposition 4.2. *If $P \overset{\alpha}{\Longrightarrow} E$ then E contains at most one occurrence of $\circ$.*

Proof. As τ-actions never introduces placeholders $\circ$. $\square$

Our labelled bisimilarity will compare pure processes; however, rules (Enter), (Co-Enter), and (Exit) result in extended processes. This means that we need a method of comparing extended processes. This will be carried out implicitly by applying them to pure processes.

Definition 4.3. *Let E, E_1, E_2 be extended processes and R be a pure process. Then,*

$$\begin{array}{rclrcl} \mathbf{0} \odot R & \overset{\text{def}}{=} & \mathbf{0} & (E_1 \mid E_2) \odot R & \overset{\text{def}}{=} & (E_1 \odot R) \mid (E_2 \odot R) \\ n[E] \odot R & \overset{\text{def}}{=} & n[E \odot R] & \boldsymbol{\nu} nE \odot R & \overset{\text{def}}{=} & \boldsymbol{\nu} n(E \odot R) \text{ if } n \notin \text{fn}(R) \\ \circ \odot R & \overset{\text{def}}{=} & R & C.E \odot R & \overset{\text{def}}{=} & C.(E \odot R) \\ !C.E \odot R & \overset{\text{def}}{=} & !C.(E \odot R)\ . & & & \end{array}$$

Definition Bisimilarity. *A symmetric relation $\mathcal{S}$ over* pure processes *is a* bisimulation *if $P \ \mathcal{S}\ Q$ implies:*

(1) If $P \xrightarrow{\alpha} P'$ then there exists Q' such that $Q \overset{\hat{\alpha}}{\Longrightarrow} Q'$ and $P' \ \mathcal{S}\ Q'$.

(2) If $P \xrightarrow{\alpha} E_1$, $\alpha \in \{\mathtt{enter}\langle n,h\rangle, \mathtt{exit}\langle n,h\rangle\}$, then for all R there exists E_2 such that $Q \overset{\alpha}{\Longrightarrow} E_2$ and $E_1 \odot R \ \mathcal{S}\ E_2 \odot R$.

(3) If $P \xrightarrow{\overline{\mathtt{enter}}\langle n,h\rangle} E_1$ then for all m and R there exists E_2 such that $Q \overset{\overline{\mathtt{enter}}\langle n,h\rangle}{\Longrightarrow} E_2$ and $E_1 \odot m[R] \ \mathcal{S}\ E_2 \odot m[R]$.

P and Q are bisimilar, written $P \approx Q$, if $P \ \mathcal{S}\ Q$ for some bisimulation $\mathcal{S}$.

The bisimilarity is defined in *early* style as the universal quantification precedes the existential one. Notice that only actions (and not pre-actions) are taken into account; moreover, in clause 1 the action α cannot belong to $\{\mathtt{enter}\langle n,h\rangle, \mathtt{exit}\langle n,h\rangle, \overline{\mathtt{enter}}\langle n,h\rangle\}$ as the derivative is a process. Note also that the definition is such that the standard proof technique can not be used to prove that it is a transitive relation. However the proof of our characterisation result does not rely on $\approx$ being transitive; but it will follow trivially from the characterisation, namely that $\approx$ coincides with the equivalence relation $\cong$.

Theorem 4.5. *The bisimilarity is contextual.*

Proof. It is straightforward to prove that $\approx$ is preserved by prefixing and iteration. We treat the other three constructs simultaneously.

Let $\mathcal{S}$ be the least symmetric relation such that:

(1) $\approx \subseteq \mathcal{S}$

(2) $P \;\mathcal{S}\; Q$ implies $P \mid R \;\mathcal{S}\; Q \mid R$ and $R \mid P \;\mathcal{S}\; R \mid Q$ for all processes R

(3) $P \;\mathcal{S}\; Q$ implies $n[P] \;\mathcal{S}\; n[Q]$

(4) $P \;\mathcal{S}\; Q$ implies $\boldsymbol{\nu} nP \;\mathcal{S}\; \boldsymbol{\nu} nQ$.

We prove that $\mathcal{S}$ is a bisimilarity up to $\equiv$[1], by induction on why two processes P and Q are in $\mathcal{S}$. The case when $P \approx Q$ follows by definition.

(1) $P \mid R \;\mathcal{S}\; Q \mid R$ because $P \;\mathcal{S}\; Q$. We now do induction on the derivation $P \mid R \xrightarrow{\alpha} E$. Most cases are straightforward. We focus on the most interesting ones, when $\alpha = \tau$ and hence E is a pure process U.

(a) $P \mid R \xrightarrow{\tau} U$ because $P \xrightarrow{\mathtt{pre\_enter}\langle n,h\rangle} \boldsymbol{\nu}\tilde{p}\langle P_1\rangle P_2$ and $R \xrightarrow{\overline{\mathtt{pre\_enter}}\langle n,h\rangle} \boldsymbol{\nu}\tilde{r}\langle R_1\rangle R_2$, with $U = \boldsymbol{\nu}\tilde{p}\boldsymbol{\nu}\tilde{r}(n[P_1 \mid R_1] \mid P_2 \mid R_2)$. By applying rule (Enter) we have

$$P \xrightarrow{\mathtt{enter}\langle n,h\rangle} E_1 = \boldsymbol{\nu}\tilde{p}(n[\circ \mid P_1] \mid P_2)$$

and $U \equiv \boldsymbol{\nu}\tilde{r}((E_1 \odot R_1) \mid R_2)$. By the inductive hypothesis, $P \;\mathcal{S}\; Q$ and for any process T there is a E_2 such that $Q \xRightarrow{\mathtt{enter}\langle n,h\rangle} E_2$ and $E_1 \odot T \equiv \mathcal{S} \equiv E_2 \odot T$. Thus

i. $P \mid R \xrightarrow{\tau} U \equiv \boldsymbol{\nu}\tilde{r}\big((E_1 \odot R_1) \mid R_2\big)$ and

ii. $Q \mid R \xrightarrow{\tau}{}^{*} Z \equiv \boldsymbol{\nu}\tilde{r}\big((E_2 \odot R_1) \mid R_2\big)$.

If we choose $T = R_1$ then we get $E_1 \odot R_1 \equiv \mathcal{S} \equiv E_2 \odot R_1$. As $\mathcal{S}$ and $\equiv$ are preserved by parallel composition and restriction, we obtain $U \equiv \mathcal{S} \equiv Z$, as required.

(b) $P \mid R \xrightarrow{\tau} U$ because $P \xrightarrow{\overline{\mathtt{pre\_enter}}\langle n,h\rangle} \boldsymbol{\nu}\tilde{p}\langle P_1\rangle P_2$, $R \xrightarrow{\mathtt{pre\_enter}\langle n,h\rangle} \boldsymbol{\nu}\tilde{r}\langle m[R_1]\rangle R_2$, with $U = \boldsymbol{\nu}\tilde{p}\boldsymbol{\nu}\tilde{q}(n[P_1 \mid m[R_1]] \mid P_2 \mid R_2)$. By applying rule (Co-Enter) we have

$$P \xrightarrow{\overline{\mathtt{enter}}\langle n,h\rangle} E_1 = \boldsymbol{\nu}\tilde{p}(n[\circ \mid P_1] \mid P_2)$$

and $U \equiv \boldsymbol{\nu}\tilde{r}\big((E_1 \odot m[R_1]) \mid R_2\big)$ (we recall that by Lemma 3.1 only ambients can migrate). By the inductive hypothesis, $P \;\mathcal{S}\; Q$ and for any m and T there is a E_2 such that $Q \xRightarrow{\overline{\mathtt{enter}}\langle n,h\rangle} E_2$ and $E_1 \odot m[T] \equiv \mathcal{S} \equiv E_2 \odot m[T]$. Thus

i. $P \mid R \xrightarrow{\tau} U \equiv \boldsymbol{\nu}\tilde{r}\big((E_1 \odot m[R_1]) \mid R_2\big)$ and

ii. $Q \mid R \xrightarrow{\tau}{}^{*} Z \equiv \boldsymbol{\nu}\tilde{r}\big((E_2 \odot m[R_1]) \mid R_2\big)$.

If we choose $T = R_1$ then we get $E_1 \odot m[R_1] \equiv \mathcal{S} \equiv E_2 \odot m[R_1]$. As $\mathcal{S}$ and $\equiv$ are preserved by parallel composition and restriction, we obtain $U \equiv \mathcal{S} \equiv Z$, as required.

[1] The soundness of the up to $\equiv$ proof technique is completely straightforward.

 (c) The other cases are either similar or straightforward and are left to the reader.

(2) $n[P] \;\mathcal{S}\; n[Q]$ because $P \;\mathcal{S}\; Q$. We do induction on why $n[P] \xrightarrow{\alpha} E$. We give details only in the cases when α is $\mathtt{enter}\langle m,h\rangle$, or $\mathtt{pop}\langle n,h\rangle$, and for the base cases of the corresponding inductions. The remaining cases, when α is τ, $\overline{\mathtt{enter}}\langle n,h\rangle$, $\mathtt{exit}\langle m,h\rangle$, or $\mathtt{free}\langle n,h\rangle$, are similar.

 (a) Let $n[P] \xrightarrow{\mathtt{enter}\langle m,h\rangle} E_1 = m[\circ \mid n[P']]$ because $P \xrightarrow{\mathtt{in}\langle m,h\rangle} P'$. For an arbitrary process Z we are required to find a move $n[Q] \xrightarrow{\mathtt{enter}\langle m,h\rangle} E_2$ such that $E_1 \odot Z \equiv \mathcal{S} \equiv E_2 \odot Z$. As $P \;\mathcal{S}\; Q$, we may use induction to find a Q' such that $Q \xRightarrow{\mathtt{in}\langle m,h\rangle} Q'$ and $P' \equiv \mathcal{S} \equiv Q'$. We may now let $E_2 = m[\circ \mid n[Q']]$ as $n[Q] \xRightarrow{\mathtt{enter}\langle m,h\rangle} E_2$ and since $\mathcal{S}$ and $\equiv$ are preserved by parallel composition and ambient constructor, we get $E_1 \odot Z \equiv m[Z \mid n[P']] \equiv \mathcal{S} \equiv m[Z \mid n[Q']] \equiv E_2 \odot Z$, as required.

 (b) Let $n[P] \xrightarrow{\mathtt{pop}\langle n,h\rangle} \boldsymbol{\nu}\tilde{p}(n[P_1] \mid P_2)$ because $P \xrightarrow{\mathtt{pre\_exit}\langle n,h\rangle} \boldsymbol{\nu}\tilde{p}\langle P_1\rangle P_2$. By applying rule (Exit)

$$P \xrightarrow{\mathtt{exit}\langle n,h\rangle} E_1 = \boldsymbol{\nu}\tilde{p}(n[\circ \mid P_1] \mid P_2)) \; .$$

 As $P \;\mathcal{S}\; Q$, by induction it holds that for all processes Z there exists $E_2 = \boldsymbol{\nu}\tilde{q}(n[\circ \mid Q_1] \mid Q_2)$ such that $Q \xRightarrow{\mathtt{exit}\langle n,h\rangle} E_2$ and $E_1 \odot Z \equiv \mathcal{S} \equiv E_2 \odot Z$. So, choosing the particular case when Z is $\mathbf{0}$ we have $n[Q] \xrightarrow{\mathtt{pop}\langle n,h\rangle} \boldsymbol{\nu}\tilde{q}(n[Q_1] \mid Q_2)$ and $\boldsymbol{\nu}\tilde{p}(n[P_1] \mid P_2) \equiv E_1 \odot \mathbf{0} \equiv \mathcal{S} \equiv E_2 \odot \mathbf{0} \equiv \boldsymbol{\nu}\tilde{q}(n[Q_1] \mid Q_2)$, as required.

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

□

Theorem Soundness. *For any (pure) processes P and Q, if $P \approx Q$ then $P \cong Q$.*

Proof. The relation $\approx$ is: (i) reduction closed, by Theorem 3.2; (ii) barb preserving, by Lemma 3.3; (iii) contextual, by Theorem 4.5. □

As a consequence, the bisimilarity represents a proof technique for reduction barbed congruence.

The next step is to prove that *the bisimilarity completely characterises reduction barbed congruence*. To this end we use a special context $\mathsf{SPY}_\alpha\langle n,h_1,h_2,[\cdot]\rangle$, parameterised on actions α, for $\alpha \in \{\mathtt{enter}\langle n,h\rangle, \mathtt{exit}\langle n,h\rangle, \overline{\mathtt{enter}}\langle n,h\rangle\}$, which allows us to *spy* on any process R plugged into the hole. This context is necessary when proving completeness to guarantee that the processes R and $m[R]$, appearing in the universal quantification of the definition of the bisimilarity, do not reduce. The context is defined so that for any process R, if $\alpha \in \{\mathtt{enter}\langle n,h\rangle, \mathtt{exit}\langle n,h\rangle\}$, then

$$\mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle \Downarrow_{\mathtt{pop}\langle n,h_i\rangle}$$

and similarly, for any name m, if $\alpha = \overline{\mathtt{enter}}\langle n, h\rangle$, then

$$m[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle] \Downarrow_{\mathtt{pop}\langle n,h_i\rangle}$$

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

We define our spying contexts using a straightforward form of *internal choice.*

Definition 4.7. *Given any (pure) processes P and Q we define*

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

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

Note that, up to structural congruence

$$P \oplus Q \xrightarrow{\tau} P \mid \boldsymbol{\nu} r(\mathtt{open}\langle r\rangle.Q)$$
$$P \oplus Q \xrightarrow{\tau} Q \mid \boldsymbol{\nu} r(\mathtt{open}\langle r\rangle.P)$$

and for virtually any behavioural equivalence and any process R, $\boldsymbol{\nu} r(\mathtt{open}\langle r\rangle.R) = \mathbf{0}$.

Definition Spy cages.

(*1*) *If $\alpha \in \{\mathtt{enter}\langle n, h\rangle,\ \mathtt{exit}\langle n, h\rangle\}$ then*

$$\mathsf{SPY}_\alpha\langle n, h_1, h_2, [\cdot]\rangle \stackrel{\text{def}}{=} \boldsymbol{\nu} z\big(\,(z[\mathtt{out}\langle n, h_1\rangle] \mid [\cdot]) \oplus (z[\mathtt{out}\langle n, h_2\rangle] \mid [\cdot])\,\big)$$

(*2*) *If $\alpha = \overline{\mathtt{enter}}\langle n, h\rangle$ then*

$$\mathsf{SPY}_\alpha\langle n, h_1, h_2, [\cdot]\rangle \stackrel{\text{def}}{=} (\mathtt{out}\langle n, h_1\rangle \mid [\cdot]) \oplus (\mathtt{out}\langle n, h_2\rangle \mid [\cdot]).$$

The above spy cages are formally *multi-hole contexts* [Sangiorgi and Walker 2001a] as the same hole occurs more than once (in this case, twice). The following result formally states the above mentioned property of the *spy cages.*

Lemma 4.9. *Let $C[\cdot]$ be a static context, R a process, n a name, and h_1, h_2 fresh names. Then:*

(*1*) *If $C[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle] \xrightarrow{\tau} P$, with $\alpha \in \{\mathtt{enter}\langle n, h\rangle,\ \mathtt{exit}\langle n, h\rangle\}$, and $P \Downarrow_{\mathtt{pop}\langle n,h_i\rangle}$, for $i \in \{1, 2\}$, then there is a static context $C'[\cdot]$ such that $P = C'[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle]$.*

(*2*) *If $C[m[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle]] \xrightarrow{\tau} P$, $\alpha{=}\overline{\mathtt{enter}}\langle n, h\rangle$, $P \Downarrow_{\mathtt{pop}\langle n,h_i\rangle}$, and $i \in \{1, 2\}$, then there is a static context $C'[\cdot]$ such that $P{=}C'[m[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle]]$.*

Proof. We prove the first the part; the second part is similar. The construction of $\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle$ assures that if $C[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle] \xrightarrow{\tau} P$, then either there is an arbitrary context C' such that $P = C'[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle]$, or $P = C[P']$ where

$$\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle \xrightarrow{\tau} P'.$$

But if $\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle \xrightarrow{\tau} P'$, then either $P \not\Downarrow_{\mathtt{pop}\langle n,h_1\rangle}$ or $P \not\Downarrow_{\mathtt{pop}\langle n,h_2\rangle}$, against the hypotheses. □

When proving the completeness of $\approx$ with respect to $\cong$, the notion of $\mathtt{pop}$ barb, i.e. the ability of a process to perform a $\mathtt{pop}$ action, will be very useful. Actually, by virtue of Theorem 3.7, our completeness result will prove that $\approx$ contains $\cong_{\mathrm{pop}}$. To this end we need a last lemma which allows us to remove the spy cages $\mathsf{SPY}_\alpha\langle n, h_1, h_2, \cdot\rangle$, up to $\cong_{\mathrm{pop}}$.

Lemma Cutting lemma. *Let $C_1[\cdot]$ and $C_2[\cdot]$ be static contexts, P, Q and R processes, and h_1 and h_2 fresh names.*

(*1*) *If $C_1[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle] \cong_{\mathrm{pop}} C_2[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle]$ then $C_1[R] \cong_{\mathrm{pop}} C_2[R]$.*
(*2*) *If $P \mid R \cong_{\mathrm{pop}} Q \mid R$, with $\mathrm{fn}(R) \cap \mathrm{fn}(P, Q) = \emptyset$, then $P \cong_{\mathrm{pop}} Q$.*

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

$$\boldsymbol{\nu}(h_1h_2)(C_1[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle]) \cong_{\mathrm{pop}} \boldsymbol{\nu}(h_1h_2)(C_2[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle]) \ .$$

As h_1 and h_2 are fresh,

$$\boldsymbol{\nu}(h_1h_2)C_i[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle] \equiv C_i[\boldsymbol{\nu}(h_1h_2)\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle]$$

for $i \in \{1, 2\}$. By exhibiting an appropriate bisimulation it is easy to prove that

$$\boldsymbol{\nu}(h_1h_2)\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle \approx R.$$

By Theorem 4.6, $\approx$ implies $\cong$; by Theorem 3.7, $\cong$ and $\cong_{\mathrm{pop}}$ coincide. These results imply that

$$C_i[\boldsymbol{\nu}(h_1h_2)\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle] \cong_{\mathrm{pop}} C_i[R]$$

for $i \in \{1, 2\}$, and hence, by transitivity, $C_1[R] \cong_{\mathrm{pop}} C_2[R]$, as required.

To prove Part 2 we set $\tilde{r} = \mathrm{fn}(R)$; it is straightforward to prove $\boldsymbol{\nu}\tilde{r}R \approx \mathbf{0}$, from which $\boldsymbol{\nu}\tilde{r}R \cong_{\mathrm{pop}} \mathbf{0}$ follows. Finally, since $\mathrm{fn}(R) \cap \mathrm{fn}(P, Q) = \emptyset$ and $\cong_{\mathrm{pop}}$ is preserved by restriction we get:

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

as required. □

Now, everything is in place to prove our main result.

Theorem Characterisation. *The bisimilarity and the reduction barbed congruence coincide.*

Proof. By Theorem 4.6 the bisimilarity is contained in the reduction barbed congruence. As to the completeness part, by Theorem 3.7, it suffices to prove that the relation

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

is a bisimilarity. We recall that

$$\begin{array}{rcl} P \oplus Q & \xrightarrow{\tau} & \boldsymbol{\nu}r(P \mid \mathtt{open}\langle r\rangle.Q \mid \mathbf{0}) \\ P \oplus Q & \xrightarrow{\tau} & \boldsymbol{\nu}r(\mathtt{open}\langle r\rangle.P \mid Q \mid \mathbf{0}) \ . \end{array}$$

As $r \notin \mathrm{fn}(P, Q)$ and $\boldsymbol{\nu}r(\mathtt{open}\langle r\rangle.R) \cong_{\mathrm{pop}} \mathbf{0}$ (the two processes are trivially bisimilar), it follows that $P \oplus Q \xrightarrow{\tau}\cong_{\mathrm{pop}} P$ and $P \oplus Q \xrightarrow{\tau}\cong_{\mathrm{pop}} Q$. In the remainder of the proof we use Lemma 3.6 without comment.

Let us first consider the three possible higher-order actions $P \xrightarrow{\alpha} E_1$:

(1) Let $P \xrightarrow{\alpha} E_1 = \boldsymbol{\nu}\tilde{p}(n[\circ \mid P_1] \mid P_2)$, with $\alpha = \mathtt{enter}\langle n,h\rangle$. We want to conclude that for all processes R there is a matching move $Q \overset{\mathtt{enter}\langle n,h\rangle}{\Longrightarrow} E_2$ such that $E_1 \odot R \;\mathcal{S}\; E_2 \odot R$. Given a process R we define:

$$C_{\alpha R}[\cdot] \overset{\text{def}}{=} [\cdot] \mid n[\overline{\mathtt{in}}\langle n,h\rangle.(\mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle \oplus a[\mathtt{out}\langle n,h_3\rangle])]$$

with a, h_i fresh. As $P \cong_{\text{pop}} Q$, it follows that $C_{\alpha R}[P] \cong_{\text{pop}} C_{\alpha R}[Q]$. So, if we define $C_1[\cdot] = \boldsymbol{\nu}\tilde{p}(n[[\cdot] \mid P_1] \mid P_2)$, then

$$\begin{array}{ll} C_{\alpha R}[P] & \\ \xrightarrow{\tau} & \boldsymbol{\nu}\tilde{p}(n[P_1 \mid (\mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle \oplus a[\mathtt{out}\langle n,h_3\rangle])] \mid P_2) \\ \xrightarrow{\tau} & \boldsymbol{\nu}\tilde{p}(n[P_1 \mid \boldsymbol{\nu}p(\mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle \mid \mathtt{open}\langle p\rangle.a[\mathtt{out}\langle n,h_3\rangle] \mid \mathbf{0})] \mid P_2) \\ \equiv & \boldsymbol{\nu}\tilde{p}(n[P_1 \mid \mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle \mid \boldsymbol{\nu}p(\mathtt{open}\langle p\rangle.a[\mathtt{out}\langle n,h_3\rangle])] \mid P_2) \\ \cong_{\text{pop}} & \boldsymbol{\nu}\tilde{p}(n[\mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle \mid P_1] \mid P_2) \\ = & C_1[\mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle] \end{array}$$

then there is a process Z such that

$$C_{\alpha R}[Q] \xrightarrow{\tau}{}^{*} Z \text{ and } C_1[\mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle] \cong_{\text{pop}} Z.$$

As a consequence, $Z{\Downarrow}_{\mathtt{pop}\langle n,h_1\rangle}$, $Z{\Downarrow}_{\mathtt{pop}\langle n,h_2\rangle}$, and $Z{\not\Downarrow}_{\mathtt{pop}\langle n,h_3\rangle}$. This implies that in the reductions sequence $C_{\alpha R}[Q] \xrightarrow{\tau}{}^{*} Z$ the prefix $\overline{\mathtt{in}}\langle n,h\rangle$ is consumed. More precisely, by Lemma 4.9(1) there exist static contexts $C'[\cdot], C''[\cdot]$ and $C_2[\cdot]$, where names a, h_1, h_2, and h_3 do not occur free, such that:

$$\begin{array}{lll} C_{\alpha R}[Q] & = & Q \mid n[\overline{\mathtt{in}}\langle n,h\rangle.(\mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle \oplus a[\mathtt{out}\langle n,h_3\rangle])] \\ & \xrightarrow{\tau}{}^{*}\xrightarrow{\tau} & C'[\mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle \oplus a[\mathtt{out}\langle n,h_3\rangle]] \\ & \xrightarrow{\tau}{}^{*}\xrightarrow{\tau} & C''[\boldsymbol{\nu}p(\mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle \mid \mathtt{open}\langle p\rangle.a[\mathtt{out}\langle n,h_3\rangle] \mid \mathbf{0})] \\ & \xrightarrow{\tau}{}^{*} & C_2[\boldsymbol{\nu}p(\mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle \mid \mathtt{open}\langle p\rangle.a[\mathtt{out}\langle n,h_3\rangle] \mid \mathbf{0})] \\ & = & Z \\ & \equiv & C_2[\mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle \mid \boldsymbol{\nu}p(\mathtt{open}\langle p\rangle.a[\mathtt{out}\langle n,h_3\rangle])] \\ & \cong_{\text{pop}} & C_2[\mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle] \end{array}$$

Let $E_2 \overset{\text{def}}{=} C_2[\circ]$. By Lemma 4.10(1), we have $E_1 \odot R = C_1[R] \cong_{\text{pop}} C_2[R] = E_2 \odot R$. It remains to show that $Q \overset{\mathtt{enter}\langle n,h\rangle}{\Longrightarrow} E_2$.
Examining the above reductions sequence from $C_{\alpha R}[Q]$ we derive that

$$Q \xrightarrow{\tau}{}^{*} \xrightarrow{\mathtt{enter}\langle n,h\rangle} C'[\circ] \xrightarrow{\tau}{}^{*} C_2[\circ],$$

and therefore we have the required corresponding action $Q \overset{\mathtt{enter}\langle n,h\rangle}{\Longrightarrow} E_2$.

(2) Let $P \xrightarrow{\alpha} E_1 = \boldsymbol{\nu}\tilde{p}(n[\circ \mid P_1] \mid P_2)$, with $\alpha = \mathtt{exit}\langle n,h\rangle$. Again we want to conclude that for all processes R there is a a matching move $Q \overset{\mathtt{exit}\langle n,h\rangle}{\Longrightarrow} E_2$ such that $E_1 \odot R \;\mathcal{S}\; E_2 \odot R$. The proof strategy is the same as in the first case except that here, given R, we use the context $C_{\alpha R}[\cdot]$ defined as

$$n[[\cdot] \mid \mathsf{SPY}_\alpha\langle n,h_1,h_2,R\rangle] \mid \overline{\mathtt{out}}\langle n,h\rangle.(a[b[\mathtt{out}\langle a,h_3\rangle]] \oplus a[b[\mathtt{out}\langle a,h_4\rangle]])$$

with a, b, h_i fresh. Again we have $C_{\alpha R}[P] \cong_{\rm pop} C_{\alpha R}[Q]$. So, if we define $C_1[\cdot] = \boldsymbol{\nu}\tilde{p}(n[[\cdot] \mid P_1] \mid P_2)$, then

$$C_{\alpha R}[P] \xrightarrow{\tau} \xrightarrow{\tau}\cong_{\rm pop} C_1[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle] \mid a[b[\mathtt{out}\langle a, h_3\rangle]]$$

then there is a process Z such that

$$C_{\alpha R}[Q] \xrightarrow{\tau}{}^{*} Z \text{ and } C_1[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle] \mid a[b[\mathtt{out}\langle a, h_3\rangle]] \cong_{\rm pop} Z.$$

As a consequence, $Z{\Downarrow}_{\mathtt{pop}\langle n,h_1\rangle}$, $Z{\Downarrow}_{\mathtt{pop}\langle n,h_2\rangle}$, $Z{\Downarrow}_{\mathtt{pop}\langle a,h_3\rangle}$, and $Z{\not\Downarrow}_{\mathtt{pop}\langle a,h_4\rangle}$. This implies that in the reductions sequence $C_{\alpha R}[Q] \xrightarrow{\tau}{}^{*} Z$ the prefix $\overline{\mathtt{out}}\langle n, h\rangle$ is consumed. More precisely, by Lemma 4.9(1) there exist static contexts $C'[\cdot], C''[\cdot]$ and $C_2[\cdot]$ where names a, b, h_i do not occur free, such that:

$$\begin{array}{rl}
C_{\alpha R}[Q] = & n[Q \mid \mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle] \mid \overline{\mathtt{out}}\langle n, h\rangle.(a[b[\mathtt{out}\langle a, h_3\rangle]] \oplus a[b[\mathtt{out}\langle a, h_4\rangle]]) \\
\xrightarrow{\tau}{}^{*} \xrightarrow{\tau} & C'[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle] \mid (a[b[\mathtt{out}\langle a, h_3\rangle]] \oplus a[b[\mathtt{out}\langle a, h_4\rangle]]) \\
\xrightarrow{\tau}{}^{*} \xrightarrow{\tau} & C''[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle] \mid \boldsymbol{\nu}p(a[b[\mathtt{out}\langle a, h_3\rangle] \mid \mathtt{open}\langle p\rangle.a[b[\mathtt{out}\langle a, h_4\rangle]] \mid \mathbf{0})] \\
\xrightarrow{\tau}{}^{*} & C_2[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle] \mid \boldsymbol{\nu}p(a[b[\mathtt{out}\langle a, h_3\rangle] \mid \mathtt{open}\langle p\rangle.a[b[\mathtt{out}\langle a, h_4\rangle]] \mid \mathbf{0})] \\
= & Z \\
\cong_{\rm pop} & C_2[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle] \mid a[b[\mathtt{out}\langle a, h_3\rangle]].
\end{array}$$

Let $E_2 \stackrel{\rm def}{=} C_2[\circ]$. By Lemmas 4.10(1) and 4.10(2), we obtain

$$E_1 \odot R = C_1[R] \cong_{\rm pop} C_2[R] = E_2 \odot R.$$

Thus, from $C_{\alpha R}[Q]$ we can derive $Q \xRightarrow{\mathtt{exit}\langle n,h\rangle} E_2$, as required.

(3) Let $P \xrightarrow{\alpha} E_1 = \boldsymbol{\nu}\tilde{p}(n[\circ \mid P_1] \mid P_2)$, with $\alpha = \overline{\mathtt{enter}}\langle n, h\rangle$. Again we need to find some E_2 such that $Q \xRightarrow{\overline{\mathtt{enter}}\langle n,h\rangle} E_2$ and $E_1 \odot m[R] \;\mathcal{S}\; E_2 \odot m[R]$. Given R, this time we use the context

$$C_{\alpha m[R]}[\cdot] \stackrel{\rm def}{=} [\cdot] \;\big|\; m[\mathtt{in}\langle n, h\rangle.(\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle \oplus \mathtt{out}\langle n, h_3\rangle)]$$

with h_i fresh. Arguing as usual from $C_{\alpha m[R]}[P] \cong_{\rm pop} C_{\alpha m[R]}[Q]$, if we define $C_1[\cdot] = \boldsymbol{\nu}\tilde{p}(n[[\cdot] \mid P_1] \mid P_2)$, we know that since

$$C_{\alpha m[R]}[P] \xrightarrow{\tau} \xrightarrow{\tau}\cong_{\rm pop} C_1[m[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle]]$$

there is a process Z such that

$$C_{\alpha m[R]}[Q] \xrightarrow{\tau}{}^{*} Z \text{ and } C_1[m[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle]] \cong_{\rm pop} Z.$$

As a consequence, $Z{\Downarrow}_{\mathtt{pop}\langle n,h_1\rangle}$, $Z{\Downarrow}_{\mathtt{pop}\langle n,h_2\rangle}$, and $Z{\not\Downarrow}_{\mathtt{pop}\langle n,h_3\rangle}$. This implies that in the reductions sequence $C_{\alpha m[R]}[Q] \xrightarrow{\tau}{}^{*} Z$ the prefix $\mathtt{in}\langle n, h\rangle$ is consumed. More precisely, this time by Lemma 4.9(2), there exist static contexts $C'[\cdot], C''[\cdot]$ and $C_2[\cdot]$, where names h_i do not occur free, such that:

$$\begin{array}{rl}
C_{\alpha m[R]}[Q] = & Q \mid m[\mathtt{in}\langle n, h\rangle.(\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle \oplus \mathtt{out}\langle n, h_3\rangle)] \\
\xrightarrow{\tau}{}^{*} \xrightarrow{\tau} & C'[m[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle \oplus \mathtt{out}\langle n, h_3\rangle]] \\
\xrightarrow{\tau}{}^{*} \xrightarrow{\tau} & C''[m[\boldsymbol{\nu}p(\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle \mid \mathtt{open}\langle p\rangle.\mathtt{out}\langle n, h_3\rangle \mid \mathbf{0})]] \\
\xrightarrow{\tau}{}^{*} & C_2[m[\boldsymbol{\nu}p(\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle \mid \mathtt{open}\langle p\rangle.\mathtt{out}\langle n, h_3\rangle \mid \mathbf{0})]] \\
= & Z \\
\cong_{\rm pop} & C_2[m[\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle]]
\end{array}$$

Let $E_2 \stackrel{\text{def}}{=} C_2[\circ]$. By Lemma 4.10(1) we have

$$E_1 \odot m[R] = C_1[m[R]] \cong_{\text{pop}} C_2[m[R]] = E_2 \odot m[R].$$

An analysis of the above reductions gives

$$Q \xrightarrow{\tau}{}^{*} \xrightarrow{\overline{\texttt{enter}}\langle n,h\rangle} C'[\circ] \xrightarrow{\tau}{}^{*} C_2[\circ] = E_2,$$

as required.

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

(4) Let $P \xrightarrow{\alpha} P'$, with $\alpha = \texttt{in}\langle n, h\rangle$. We want to conclude that there is Q' such that $Q \xRightarrow{\texttt{in}\langle n,h\rangle} Q'$ and $P' \;\mathcal{S}\; Q'$. We define:

$$C_\alpha[\cdot] \stackrel{\text{def}}{=} a[[\cdot] \mid \texttt{out}\langle n, h_1\rangle.\overline{\texttt{open}}\langle a\rangle] \;\big|\; n[\overline{\texttt{in}}\langle n, h\rangle] \;\big|\; \overline{\texttt{out}}\langle n, h_1\rangle.\texttt{open}\langle a\rangle.(h_2 \oplus h_3)$$

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

$$\begin{array}{lll}
C_\alpha[P] & \xrightarrow{\tau} & n[a[P' \mid \texttt{out}\langle n, h_1\rangle.\overline{\texttt{open}}\langle a\rangle]] \;\big|\; \overline{\texttt{out}}\langle n, h_1\rangle.\texttt{open}\langle a\rangle.(h_2 \oplus h_3) \\
 & \xrightarrow{\tau} & n[\,] \;\big|\; a[P' \mid \overline{\texttt{open}}\langle a\rangle] \;\big|\; \texttt{open}\langle a\rangle.(h_2 \oplus h_3) \\
 & \xrightarrow{\tau} & n[\,] \;\big|\; P' \;\big|\; \mathbf{0} \;\big|\; (h_2 \oplus h_3) \\
 & \xrightarrow{\tau}\cong_{\text{pop}} & P' \;\big|\; f[\boldsymbol{\nu} z(z[\texttt{out}\langle f, h_2\rangle])]
\end{array}$$

(notice that $n[\,] \cong_{\text{pop}} \mathbf{0}$, see Section 6) then there is a process Z such that

$$C_\alpha[Q] \xrightarrow{\tau}{}^{*} Z \text{ and } P' \mid f[\boldsymbol{\nu} z(z[\texttt{out}\langle f, h_2\rangle])] \cong_{\text{pop}} Z.$$

As a consequence, $Z \Downarrow_{\texttt{pop}\langle f,h_2\rangle}$ and $Z \not\Downarrow_{\texttt{pop}\langle f,h_3\rangle}$. This implies that in the reductions sequence $C_\alpha[Q] \xrightarrow{\tau}{}^{*} Z$ the whole context $C_\alpha[\cdot]$ is consumed (up to $\cong_{\text{pop}}$) except for $f[\boldsymbol{\nu} z(z[\texttt{out}\langle f, h_2\rangle])]$. More precisely, as $n[\,] \cong_{\text{pop}} \mathbf{0}$, there exist Q_1, Q_2, Q_3, Q' such that:

$$\begin{array}{l}
C_\alpha[Q] = a[Q \mid \texttt{out}\langle n, h_1\rangle.\overline{\texttt{open}}\langle a\rangle] \;\big|\; n[\overline{\texttt{in}}\langle n, h\rangle] \;\big|\; \overline{\texttt{out}}\langle n, h_1\rangle.\texttt{open}\langle a\rangle.(h_2 \oplus h_3) \\
\xrightarrow{\tau}{}^{*} \xrightarrow{\tau} n[a[Q_1 \mid \texttt{out}\langle n, h_1\rangle.\overline{\texttt{open}}\langle a\rangle]] \;\big|\; \overline{\texttt{out}}\langle n, h_1\rangle.\texttt{open}\langle a\rangle.(h_2 \oplus h_3) \\
\xrightarrow{\tau}{}^{*} \xrightarrow{\tau} n[\,] \;\big|\; a[Q_2 \mid \overline{\texttt{open}}\langle a\rangle] \;\big|\; \texttt{open}\langle a\rangle.(h_2 \oplus h_3) \\
\xrightarrow{\tau}{}^{*} \xrightarrow{\tau} n[\,] \mid Q_3 \mid (h_2 \oplus h_3) \\
\quad \xrightarrow{\tau}{}^{*} n[\,] \mid Q' \mid \boldsymbol{\nu} p(f[\boldsymbol{\nu} z\, z[\texttt{out}\langle f, h_2\rangle]] \mid \texttt{open}\langle p\rangle.f[\boldsymbol{\nu} zz[\texttt{out}\langle f, h_3\rangle]] \mid \mathbf{0}) \\
\qquad = Z \\
\quad \cong_{\text{pop}} Q' \mid f[\boldsymbol{\nu} z(z[\texttt{out}\langle f, h_2\rangle])]
\end{array}$$

where $Q \xrightarrow{\tau}{}^{*} \xrightarrow{\texttt{in}\langle n,h\rangle} Q_1 \xrightarrow{\tau}{}^{*} Q_2 \xrightarrow{\tau}{}^{*} Q_3 \xrightarrow{\tau}{}^{*} Q'$. By Lemma 4.10(2) we obtain $P' \cong_{\text{pop}} Q'$, as required.

(5) Let $P \xrightarrow{\alpha} P'$, with $\alpha = \overline{\texttt{in}}\langle n, h\rangle$. Again we need to find some Q' such that $Q \xRightarrow{\overline{\texttt{in}}\langle n,h\rangle} Q'$ and $P' \;\mathcal{S}\; Q'$. The argument is the same as in the previous case, this time using the context $C_\alpha[\cdot]$ defined as

$$n[[\cdot] \mid \overline{\texttt{open}}\langle n, h_1\rangle.(h_2 \oplus h_3)] \;\big|\; a[\texttt{in}\langle n, h\rangle.\texttt{out}\langle n, h_4\rangle] \;\big|\; \mid \overline{\texttt{out}}\langle n, h_4\rangle.\texttt{open}\langle n, h_1\rangle$$

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

$$\begin{aligned}
C_\alpha[P] &\xrightarrow{\tau} n[a[\mathtt{out}\langle n, h_4\rangle] \mid P' \mid \overline{\mathtt{open}}\langle n, h_1\rangle.(h_2 \oplus h_3)] \\
&\qquad \Big| \; \overline{\mathtt{out}}\langle n, h_4\rangle.\mathtt{open}\langle n, h_1\rangle \\
&\xrightarrow{\tau} n[P' \mid \overline{\mathtt{open}}\langle n, h_1\rangle.(h_2 \oplus h_3)] \;\Big|\; a[\,] \;\Big|\; \mathtt{open}\langle n, h_1\rangle \\
&\xrightarrow{\tau} P' \mid (h_2 \oplus h_3) \;\Big|\; a[\,] \\
&\xrightarrow{\tau} P' \;\Big|\; f[\boldsymbol{\nu} z(z[\mathtt{out}\langle f, h_2\rangle])] \;\Big|\; a[\,] \\
&\cong_{\text{pop}} P' \;\Big|\; f[\boldsymbol{\nu} z(z[\mathtt{out}\langle f, h_2\rangle])]
\end{aligned}$$

we know that there is a process Z such that

$$C_\alpha[Q] \xrightarrow{\tau}{}^{*} Q' \text{ and } P' \mid f[\boldsymbol{\nu} z(z[\mathtt{out}\langle f, h_2\rangle])] \cong_{\text{pop}} Z.$$

As a consequence, $Z \Downarrow_{\mathtt{pop}\langle f, h_2\rangle}$ whereas $Z \not\Downarrow_{\mathtt{pop}\langle f, h_3\rangle}$. This implies that in the reduction sequence $C_\alpha[Q] \xrightarrow{\tau}{}^{*} Z$ the whole context $C_\alpha[\cdot]$ is consumed (up to $\cong_{\text{pop}}$) except for $f[\boldsymbol{\nu} z(z[\mathtt{out}\langle f, h_2\rangle])]$. More precisely, as $a[\,] \cong_{\text{pop}} \mathbf{0}$, there exist Q_1, Q_2, and Q_3 such that:

$$\begin{aligned}
C_\alpha[Q] = {}& n[Q \mid \overline{\mathtt{open}}\langle n, h_1\rangle.(h_2 \oplus h_3)] \;\Big|\; a[\mathtt{in}\langle n, h\rangle.\mathtt{out}\langle n, h_4\rangle] \;\Big| \\
& \overline{\mathtt{out}}\langle n, h_4\rangle.\mathtt{open}\langle n, h_1\rangle \\
\xrightarrow{\tau}{}^{*} \xrightarrow{\tau} {}& n[Q_1 \mid \overline{\mathtt{open}}\langle n, h_1\rangle.(h_2 \oplus h_3) \;\Big|\; a[\mathtt{out}\langle n, h_4\rangle]] \;\Big|\; \overline{\mathtt{out}}\langle n, h_4\rangle.\mathtt{open}\langle n, h_1\rangle \\
\xrightarrow{\tau}{}^{*} \xrightarrow{\tau} {}& n[Q_2 \mid \overline{\mathtt{open}}\langle n, h_1\rangle.(h_2 \oplus h_3)] \mid a[\,] \mid \mathtt{open}\langle n, h_1\rangle \\
\xrightarrow{\tau}{}^{*} \xrightarrow{\tau} {}& Q_3 \mid (h_2 \oplus h_3) \mid a[\,] \\
\xrightarrow{\tau}{}^{*} \xrightarrow{\tau} {}& Q' \mid \boldsymbol{\nu} p(f[\boldsymbol{\nu} z\, z[\mathtt{out}\langle f, h_2\rangle] \mid \mathtt{open}\langle p\rangle.f[\boldsymbol{\nu} z\, z[\mathtt{out}\langle f, h_3\rangle]]] \mid \mathbf{0}) \mid a[\,] \\
= {}& Z \\
\cong_{\text{pop}} {}& Q' \mid f[\boldsymbol{\nu} z(z[\mathtt{out}\langle f, h_2\rangle])]
\end{aligned}$$

where $Q \xrightarrow{\tau}{}^{*} \xrightarrow{\overline{\mathtt{in}}\langle n,h\rangle} Q_1 \xrightarrow{\tau}{}^{*} Q_2 \xrightarrow{\tau}{}^{*} Q_3 \xrightarrow{\tau}{}^{*} Q'$. By Lemma 4.10(2) we can conclude that $P' \cong_{\text{pop}} Q'$, as required.

(6) The six remaining cases are similar, except that we need an appropriate context. These are detailed as follows:

(a) for $\alpha = \mathtt{pop}\langle n, h\rangle$ use

$$C_\alpha[\cdot] \overset{\text{def}}{=} [\cdot] \;\Big|\; \overline{\mathtt{out}}\langle n, h\rangle.(h_1 \oplus h_2)$$

with h_1 and h_2 fresh.

(b) for $\alpha = \overline{\mathtt{out}}\langle n, h\rangle$ use

$$C_\alpha[\cdot] \overset{\text{def}}{=} [\cdot] \;\Big|\; n[a[\mathtt{out}\langle n, h\rangle.\overline{\mathtt{open}}\langle a\rangle]] \;\Big|\; \mathtt{open}\langle a\rangle.(h_1 \oplus h_2)$$

with a, h_1 and h_2 fresh.

(c) for $\alpha = \mathtt{out}\langle n, h\rangle$ use

$$C_\alpha[\cdot] \overset{\text{def}}{=} n[a[[\cdot] \mid \overline{\mathtt{open}}\langle a\rangle]] \;\Big|\; \overline{\mathtt{out}}\langle n, h\rangle.\mathtt{open}\langle a\rangle.(h_1 \oplus h_2)$$

with a, h_1 and h_2 fresh.

(d) for $\alpha = \mathtt{open}\langle n, h\rangle$ use

$$C_\alpha[\cdot] \overset{\text{def}}{=} [\cdot] \;\Big|\; n[\overline{\mathtt{open}}\langle n, h\rangle.(h_1 \oplus h_2)]$$

with h_1 and h_2 fresh.

(e) for $\alpha = \overline{\texttt{open}}\langle n, h\rangle$ use

$$C_\alpha[\cdot] \stackrel{\text{def}}{=} n[[\cdot]] \;\big|\; \texttt{open}\langle n, h\rangle.(h_1 \oplus h_2)$$

with h_1 and h_2 fresh.

(f) for $\alpha = \texttt{free}\langle n, h\rangle$ use

$$C_\alpha[\cdot] \stackrel{\text{def}}{=} [\cdot] \;\big|\; \texttt{open}\langle n, h\rangle.(h_1 \oplus h_2)$$

with h_1 and h_2 fresh.

□

Remark The role of passwords. *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 must send it an ambient which contains a fresh password. Probing for this fresh password ensures that the ambient we have sent has indeed been received at its destination. Without fresh passwords, and the new semantics for $\overline{\texttt{out}}$, there would be no distinguishing feature of the ambient sent which could be used in the probe to ensure that that ambient has indeed been received.*

Remark The role of $\overline{\texttt{out}}$. *Note also that our rules for $\overline{\texttt{out}}$, different from those in [Levi and Sangiorgi 2000], have played a crucial role in the distinguishing contexts for both $\overline{\texttt{enter}}$ and $\overline{\texttt{in}}$. The alternative semantics for $\overline{\texttt{out}}\langle n\rangle$ given in [Levi and Sangiorgi 2000] uses an auxiliary action ?n for which it is difficult to conceive of a distinguishing context.*

Remark 4.14. *The proof that the bisimilarity coincides with reduction barbed congruence relies on the fact that $\equiv$ is transitive; this is necessary to conclude that a bisimulation up to $\equiv$ is actually a bisimulation. But we did not require the transitivity of $\approx$.*

Corollary 4.15. *The bisimilarity is an equivalence relation.*

Proof. Because the bisimilarity coincides with reduction barbed congruence which is an equivalence relation. □

5. ADDING COMMUNICATION

Both Mobile Ambients, [Cardelli and Gordon 2000], and Safe Ambients, [Levi and Sangiorgi 2000], allow local communication inside ambients. The basic idea is to have an *output process* such as $\langle W\rangle$, which outputs the message W, and an input process such as $(x).Q$, which on receiving a message binds it to x in Q which then executes. The basic reduction rule therefore takes the form

$$(x).Q \mid \langle W\rangle \quad\to\quad Q\{W/x\} \mid P.$$

where $\{W\,/\,x\}$ denotes standard substitution of variable x with W, avoiding name and variable captures. In this section we show that our results can be extended to a message-passing setting.

The syntax of the message-passing SAP is given in Table VIII. 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:

Names:		$n, h, \ldots \in \mathbf{N}$	
Variables:		$x, y, \ldots \in \mathbf{X}$	
Values:			
W	$::=$	x	variable
	$\mid$	C	capability
	$\mid$	$W_1.W_2$	path
	$\mid$	ϵ	empty path
Guards:			
G	$::=$	W	expression
	$\mid$	(x)	input
	$\mid$	$\langle W \rangle$	output
Message-passing processes:			
P, Q, R	$::=$	$\mathbf{0}$	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:			
K	$::=$	$\boldsymbol{\nu} \tilde{p} \langle E \rangle F$	movement concretion
	$\mid$	$\boldsymbol{\nu} \tilde{p} \langle W \rangle E$	buffer concretion
Extended message-passing processes:			
E, F	$::=$	$\circ$	placeholder
	$\mid$	$\boldsymbol{\{} W \boldsymbol{\}}$	buffer containing W
	$\mid$	$\mathbf{0}$	nil process
	$\mid$	$E_1 \mid E_2$	parallel composition
	$\mid$	$\boldsymbol{\nu} n E$	restriction
	$\mid$	$G.E$	prefixing
	$\mid$	$n[E]$	ambient
	$\mid$	$!G.E$	replication

Table VIII. The Message-passing Calculus SAP

—$W.P$, a direct generalisation of $C.P$. Here W is any *path*, or sequence, of capabilities. These paths will be the messages allowed in our systems.

—$\langle W \rangle.P$, representing the synchronous output of the message W; the process P can not be executed until the message W has been consumed. As discussed in [Castagna and Zappa Nardelli 2002; Bugliesi et al. 2001] this is not unrealistic because communication is always local.

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

We now have variables in our language, with the construct $(x).Q$ binding x in Q. This gives rise in the standard manner to the notions of free and bound

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

$$(\text{Path})\ \frac{W_1.(W_2.E) \xrightarrow{\alpha} F}{(W_1.W_2).E \xrightarrow{\alpha} F} \qquad (\tau\ \text{Eps})\ \frac{-}{\epsilon.E \xrightarrow{\tau} E}$$

$$(\tau\ \text{Comm})\ \frac{E \xrightarrow{\langle - \rangle} \boldsymbol{\nu}\tilde{p}\langle W\rangle E' \quad F \xrightarrow{(W)} F' \quad \text{fn}(F)\cap\{\tilde{p}\}=\emptyset}{\begin{array}{l} E \mid F \xrightarrow{\tau} \boldsymbol{\nu}\tilde{p}(E' \mid F') \\ F \mid E \xrightarrow{\tau} \boldsymbol{\nu}\tilde{p}(F' \mid E') \end{array}}$$

$$(\text{Output})\ \frac{E \xrightarrow{\langle - \rangle} \boldsymbol{\nu}\tilde{p}\langle W\rangle E'}{E \xrightarrow{\langle \texttt{output} \rangle} \boldsymbol{\nu}\tilde{p}(\boldsymbol{\{}W\boldsymbol{\}} \mid E')}$$

Table IX. Labelled Transition System - Communication

variables, fv(·) and bv(·), α-equivalence and *substitutions* in which free occurrences of variables are not captured; we avoid spelling out the details. A process E is said *closed* if $\text{fv}(E) = \emptyset$; otherwise is said *open*.

The *labelled transition system* is enriched by introducing new rules for input and output transitions, and enlarging the extended processes with a new construct $\boldsymbol{\{}W\boldsymbol{\}}$ to denote a buffer containing the value W (see Table VIII):

—$E \xrightarrow{(W)} F$ means that the process E may receive the message W and continue as F

—$E \xrightarrow{\langle - \rangle} \boldsymbol{\nu}\tilde{p}\langle W\rangle F$ means that E may send the message W which shares the bound names $\tilde{p}$ with the residual F

—$E \xrightarrow{\langle \texttt{output} \rangle} \boldsymbol{\nu}\tilde{p}(\boldsymbol{\{}W\boldsymbol{\}} \mid F)$ means that E sends the message W which shares the bound names $\tilde{p}$ with the residual F.

In Table IX we give the defining rules for the operational semantics of these constructs, which should be added to those of Tables V to obtain the *lts* $E \xrightarrow{\alpha} F$ for *closed processes*. The rules are straightforward and require no comment except for action $\langle$`output`$\rangle$ which does not have structural rules, as for `enter`, $\overline{\texttt{enter}}$, and `exit`.

In order to generalise our *labelled bisimilarity* to the message-passing calculus, we extend Definition 4.3 to buffers. Let R be any open pure process such that $\text{fv}(R) = \{x\}$; intuitively here x represents the variable for receiving the message W emitted via an output action $E \xrightarrow{\langle \texttt{output} \rangle} \boldsymbol{\nu}\tilde{p}(\boldsymbol{\{}W\boldsymbol{\}} \mid F)$. Then, we define

$$\boldsymbol{\{}W\boldsymbol{\}} \odot R \stackrel{\text{def}}{=} R\{^W\!/_x\}.$$

The definition of *bisimilarity* $\approx$ must be extended by adding the following clause for outputs (the input case is included in the first-order clause of the bisimilarity):

4 If $P \xrightarrow{\langle \texttt{output} \rangle} E_1$ then for any open process R such that fv(R)=$\{x\}$ there is a extended closed process E_2 such that $Q \xRightarrow{\langle \texttt{output} \rangle} E_2$ and $E_1 \odot R \ \mathcal{S}\ E_2 \odot R$.

We now outline how to extend Theorem 4.11 to this setting, relating the bisimilarity to the reduction barbed congruence. First let us be quite precise as to how Definition 2.5 extends to our message-passing language.

Definition Reduction barbed congruence. *Reduction barbed congruence, written $\cong_{\mathrm{o}}$ is the largest symmetric relation over* message-passing pure processes *which*

—*is contextual*

—*when restricted to closed processes is reduction closed*

—*when restricted to closed processes is barb preserving.*

Defined in this manner there is an immediate mismatch between $\cong_{\mathrm{o}}$ and the bisimilarity $\approx$; the former is also defined for open terms while the latter only applies to closed terms. However we can rectify this by generalising $\approx$ to open terms in the standard manner.

Definition 5.2. *For any two pure processes P, Q in the message-passing SAP we write $P \approx_{\mathrm{o}} Q$ if for all closing substitutions σ, mappings from variables to names, we have $P\sigma \approx Q\sigma$.*

Theorem 5.3. *The relation $\approx_{\mathrm{o}}$ is contextual in the message-passing SAP.*

Proof. A straightforward extension of Theorem 4.5 considering each operator in turn. For example to show that it is preserved by input prefixing it is sufficient to show that

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

for all processes such that $\mathrm{fv}(P) \cup \mathrm{fv}(Q) \subseteq \{x\}$. However the hypothesis says that $P\{^W\!/_x\} \approx Q\{^W\!/_x\}$ for arbitrary messages W which is all that is required to prove the conclusion; the only possible moves from $(x).P$ are of the form $(x).P \xrightarrow{(W)} P\{^W\!/_x\}$.

For all other operators it is sufficient to consider only closed terms and so the reasoning is very similar to that in Theorem 4.5. The main novelty consists in using the communication rule (τ Comm) to prove that $\approx_{\mathrm{o}}$ is preserved by parallel composition. □

This, together with the straightforward extension of Lemma 3.3 to the message-passing calculus, immediately establishes that $\approx_{\mathrm{o}}$ is contained in $\cong_{\mathrm{o}}$. In fact the converse is also true.

Theorem 5.4. *Relations $\cong_{\mathrm{o}}$ and $\approx_{\mathrm{o}}$ coincide in the message-passing SAP.*

Proof. For *closed processes* the proof that $\cong_{\mathrm{o}}$ is contained in $\approx_{\mathrm{o}}$ follows from a straightforward extension of Theorem 3.7 to the message-passing calculus. It suffices to prove that the relation

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

is a bisimilarity, where $\cong_{\mathrm{pop}}$ is defined over open terms. The main difference with respect to the proof of Theorem 4.11 is that we have to consider the cases for input and output actions.

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

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

with h_1 and h_2 fresh.

(2) Let $P \xrightarrow{\langle \mathtt{output}\rangle} E_1$; we want to conclude that for any open term R such that $\mathrm{fv}(R)=\{x\}$ there is E_2 such that $Q \overset{\langle \mathtt{output}\rangle}{\Longrightarrow} E_2$ and $E_1 \odot R \;\mathcal{S}\; E_2 \odot R$. As a distinguishing context take:

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

with a, b, h_i fresh, and $\beta \in \{\mathtt{enter}\langle n, h\rangle, \mathtt{exit}\langle n, h\rangle\}$.

So we can conclude that for closed processes $P \cong_{\mathrm{o}} Q$ implies $P \approx_{\mathrm{o}} Q$.

Now consider two arbitrary *open processes* P, Q such that $P \cong_{\mathrm{o}} Q$. We need to show that $P\sigma \approx Q\sigma$ for any closing substitution σ. Let $x_1, x_2, \ldots, x_n$ be all the variables free in both P and Q. From $P \cong_{\mathrm{o}} Q$ we know that $(x_1).(x_2).\ldots.(x_n).P \cong_{\mathrm{o}} (x_1).(x_2).\ldots.(x_n).Q$ and since these are closed terms we can conclude that

$$(x_1).(x_2).\ldots.(x_n).P \approx (x_1).(x_2).\ldots.(x_n).Q\ .$$

But now examining the behaviour of these processes with respect to the input actions $(\sigma(x_1)), (\sigma(x_2)), \ldots$ we can conclude that $P\sigma \approx Q\sigma$ □

We end this section with two comments.

Notice that we have a more restricted form of message than in [Cardelli and Gordon 2000; Levi and Sangiorgi 2000]. In particular we do not allow ambient names to be transmitted. This has been a deliberate choice as, apriori, when the name is transmitted the recipient gets considerable control over that ambient. Moreover, much of the power of name transmission can still be captured in our language.

The second comment regards the formalisation of output actions using $\boldsymbol{\nu}\tilde{n}\langle M\rangle E$. As these actions are higher-order, the reader might think of avoiding the universal quantification over R by using a simpler rule for output actions such as

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

Notice that with the rule above we would be obliged to introduce bound output actions of the form $\xrightarrow{\boldsymbol{\nu}n\langle \mathtt{in}\langle n,h\rangle\rangle}$. However, such actions *would not be observable* as there is no context which is capable of recognising whether or not a process can perform them. Intuitively, this is because in the action $\xrightarrow{\boldsymbol{\nu}n\langle \mathtt{in}\langle n,h\rangle\rangle}$ the name n is private and no context may use n as an ambient name to recognise the action. This would make a serious problem when proving that the bisimilarity is contained in reduction barbed congruence.

6. EXAMPLES

SAP is an expressive language. Roughly speaking, we can consider Levi and Sangiorgi's Safe Ambients as a sub-language of SAP, up to the different operational semantics for $\overline{\texttt{out}}$. However, due to this difference there is no obvious encoding of MA (or SA) into SAP. To better understand this statement, consider Levi and Sangiorgi's encoding of MA into SA. Basically, every MA-process $n[P]$ is translated into a SA-process $\widehat{n[P]}$, where ambient n contains the encoding of P, i.e. $\widehat{P}$, in parallel with the replicated co-capabilities $!\overline{\texttt{in}}\langle n\rangle$, $!\overline{\texttt{out}}\langle n\rangle$, and $!\overline{\texttt{open}}\langle n\rangle$. Notice that all co-capabilities are within the ambient n, and migrate with it. Now, it should be evident that in a similar encoding of MA into SAP the $!\overline{\texttt{out}}\langle n\rangle$ capability should not be (statically) placed inside n, but rather in any ambient to which n can migrate. This makes the encoding quite difficult, especially when dealing with private ambients.

In the sequel we will outline how our results could form the basis for reasoning techniques for ambients. Some examples programmed in [Levi and Sangiorgi 2000] will be analysed using our bisimulations. We will also give two examples, similar to those given in [Cardelli and Gordon 2000; Levi and Sangiorgi 2000], which show that the existence of passwords can be of help when designing ambients. Although, we do not claim that passwords add extra expressive power, they are a very useful programming feature.

Routable packets. In [Cardelli and Gordon 2000], Cardelli and Gordon present a protocol to route a packet to various destinations. The content P and the destination W are contained in an ambient *route*. The act of sending P to destination W is realised by the following steps. Ambient *route* enters inside the packet and is opened. This liberates a message $\langle W\rangle$, which is then consumed so that the path W can be executed. At the end, the packet, which contains P, has reached the destination. Here is the program in Mobile Ambients:

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

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

As already pointed out in [Levi and Sangiorgi 2000], the protocol above works only under severe constraints on both 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 [Levi and Sangiorgi 2000] by providing a new protocol along the lines of the taxi protocol in [Cardelli and Gordon 1999]. Below we adapt Levi and Sangiorgi's protocol making use of passwords. We replace $\langle P, W\rangle$ with $\langle P, W, k\rangle$ 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 [Cardelli and Gordon 2000], to avoid interferences from P on the path to follow, we enclose P in an ambient d.

$$\begin{aligned} PKT &\stackrel{\mathrm{def}}{=} \; !p[\,\overline{\mathtt{in}}\langle p\rangle.\mathtt{open}\langle r\rangle.(x).x\,] \\ \langle P, W, k\rangle &\stackrel{\mathrm{def}}{=} (\boldsymbol{\nu} d)r[\,\mathtt{in}\langle p\rangle.\overline{\mathtt{open}}\langle r\rangle.\langle W.\mathtt{open}\langle d\rangle\rangle.d[\overline{\mathtt{open}}\langle d\rangle.\overline{\mathtt{open}}\langle p, k\rangle.P]\,] \end{aligned}$$

Notice that in our protocol, unlike [Cardelli and Gordon 2000; Levi and Sangiorgi 2000], 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 W. 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 [Cardelli and Gordon 2000] for controlling accesses through a firewall. Again our version is inspired by that in [Levi and Sangiorgi 2000] but now passwords are used. Ambient f represents the firewall and h_f is the password to cross it; ambient a represents a trusted agent inside which is a process Q that is supposed to cross the firewall. h_a is the password to access a. The firewall sends into the agent a pilot ambient k with the ability $\mathtt{in}\langle f, h_f\rangle$ to enter the firewall. The agent acquires the capability by opening k and then enters f. The process Q carried by the agent is finally liberated inside the firewall by the opening of ambient a. Here is the protocol:

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

Note that here, unlike [Levi and Sangiorgi 2000], the names f an a, of the firewall and agent respectively, can be considered public information; the security of the system resides in keeping the passwords h_f and h_a private.

Algebraic Laws. We now turn our attention to some algebraic laws which we can justify straightforwardly using bisimulations. In [Levi and Sangiorgi 2000] it is shown that by establishing a set of basic set of such laws between ambients non-trivial reasoning can be carried out. Indeed most of our laws are taken directly from that paper, or are simple modifications thereof. Here we show how they can be established using bisimulations, rather than the more complicated contextual reasoning in [Levi and Sangiorgi 2000]. The simplest example is

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

These two processes are bisimilar because the relation

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

is a trivial bisimulation; neither side can perform any action. Notice that this law is not true in MA. On the contrary, an important law of MA, the *perfect firewall equation*

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

where $n \notin \mathrm{fn}(P)$, is not true in our setting, nor does it hold for the Safe Ambients of [Levi and Sangiorgi 2000]. In fact, consider the case when P is given by

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

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

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

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

$$n_1[n_2[P]] \approx \mathbf{0} \qquad \text{with } n_1 \notin \mathrm{fn}(P)$$

Again, it suffices to prove that

$$\{(n_1[n_2[P]]\,,\,\mathbf{0}) \mid n_1 \notin \mathrm{fn}(P)\} \cup \{(\mathbf{0}\,,\,n_1[n_2[P]]) \mid n_1 \notin \mathrm{fn}(P)\}$$

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

Here are a collection of laws, most of which are taken from [Levi and Sangiorgi 2000].

THEOREM 6.1.

(*1*) $\boldsymbol{\nu} h(m[\mathtt{in}\langle n,h\rangle.P] \mid n[\overline{\mathtt{in}}\langle n,h\rangle.Q]) \approx \boldsymbol{\nu} h(n[Q \mid m[P]])$
(*2*) $k[m[\mathtt{in}\langle n,h\rangle.P] \mid n[\overline{\mathtt{in}}\langle n,h\rangle.Q])] \approx k[n[Q \mid m[P]]]$
(*3*) $\boldsymbol{\nu} h(\mathtt{open}\langle m,h\rangle.P \mid m[\overline{\mathtt{open}}\langle m,h\rangle.Q]) \approx \boldsymbol{\nu} h(P \mid Q)$
(*4*) $k[\mathtt{open}\langle m,h\rangle.P \mid m[\overline{\mathtt{open}}\langle m,h\rangle.Q]] \approx k[P \mid Q]$
(*5*) $\boldsymbol{\nu} h(n[m[\mathtt{out}\langle n,h\rangle.Q]] \mid \overline{\mathtt{out}}\langle n,h\rangle.P) \approx \boldsymbol{\nu} h(m[Q] \mid P)$
(*6*) $k[n[m[\mathtt{out}\langle n,h\rangle.Q]] \mid \overline{\mathtt{out}}\langle n,h\rangle.P] \approx k[m[Q] \mid P]$
(*7*) $n[\prod_i G_i.P_i] \approx \mathbf{0}$ *if* $G_i \in \{\langle\tilde{W}\rangle, \mathtt{open}\langle n,h\rangle, \overline{\mathtt{out}}\langle n,h\rangle\}$*, for all* i
(*8*) $n[\prod_i G_i.P_i] \approx \mathbf{0}$ *if* $G_i \in \{(\tilde{x}), \mathtt{open}\langle n,h\rangle, \overline{\mathtt{out}}\langle n,h\rangle\}$*, for all* i
(*9*) $n[\langle W\rangle.P \mid (x).Q] \approx n[P \mid Q\{{}^{W}\!/\!_{x}\}]$.

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

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

where LHS, RHS denote the left hand side, right hand side respectively of the identity. In the proof of parts 5 and 6 we require the law $n[\,] \approx \mathbf{0}$. □

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

THEOREM 6.2. *If* $h_a \notin \mathrm{fn}(P)$ *and* $h_f \notin \mathrm{fn}(Q)$*, then:*

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

PROOF. Similar to the proof of Equation (15) of [Levi and Sangiorgi 2000], but now applying Laws 5, 1, 4, 9, 1, 4 of Theorem 6.1. □

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

In this paper we have developed a semantic theory of SAP, a variant of Levi and Sangiorgi's Safe Ambients where each capability may contain two components: the target ambient name, and a second name which can be looked upon as either a password or a port. This second component enhances the access protocol with a form of validation for the credentials of incoming ambients as a preliminary step to a registration protocol. An example of the practical relevance and the strong intuition behind this mechanism is the negotiation of credentials that takes place when connecting to a (wireless) LAN using DHCP, or to an ISP using PPP. Distributed calculi that use an extra component to access administrative domains are quite common in the literature. For instance, in SafeDpi [Hennessy et al. 2003] parametrised code may be sent between sites using ports, which are essentially higher-order channels. A similar idea has already been used in the Seal Calculus [Castagna and Zappa Nardelli 2002]. In NBA [Bugliesi et al. 2005], a disciplined version of Boxed Ambients [Bugliesi et al. 2001], passwords play a central role in the definition of a type system; essentially, the type of a passwords records information about those ambients using that password. Finally, the notion of port in distributed calculi has also been investigated in the Channel Boxed Ambients [Phillips 2005] for which there already exists a distributed implementation [Phillips et al. 2004].

The paper focuses on bisimulation-based behavioural equivalences and more precisely on *reduction barbed congruence*, a slight variant of Milner and Sangiorgi's *barbed congruence* [Milner and Sangiorgi 1992] also called *open barbed bisimilarity* [Sangiorgi and Walker 2001b]. Reduction barbed congruence, although formulated with an equational approach, was first studied by Honda and Yoshida [1995] for the π-calculus under the name of *maximum sound theory* .

The main results of the paper are

—an lts-based operational semantics for SAP

—a labelled bisimilarity which coincides with reduction barbed congruence

—a set of algebraic laws proved using our notion of bisimilarity.

A theory of Morris-style contextual equivalence for Mobile Ambients has been already developed by Gordon and Cardelli in [Gordon and Cardelli 2002]. However, although the theory is equipped with a context lemma which allows to consider only contexts of a particular form, we believe that the verification of algebraic laws still remain quite complicated. As an example the proof of the perfect firewall equation in [Gordon and Cardelli 2002] is quite involved. In the same manner, the proofs in [Levi and Sangiorgi 2000] of the algebraic laws for Safe Ambients using contextual reasonings are definitely more complicated than our single-pair bisimulation proofs.

Higher-order labelled transition systems for Mobile Ambients can be found in [Cardelli and Gordon 1996; Gordon and Cardelli 2002; Vigliotti 1999; Ferrari et al. 2001; Merro and Zappa Nardelli 2003]. A simple first-order lts for MA without restriction is proposed by Sangiorgi in [Sangiorgi 2001]. Using this lts the author defines an *intensional* bisimilarity for MA that separates terms on the basis of their internal structure.

Our labelled transition system is inspired by that of [Levi and Sangiorgi 2000] for Safe Ambients. The main difference is the treatment of the co-capability $\overline{\texttt{out}}$; here is exercised by the target computation space and not by the surrounding ambient; this allows us

(1) to avoid the action $?n$ of [Levi and Sangiorgi 2000] for which it is difficult to conceive of a distinguishing context;

(2) to have simpler proof of the contextuality of the bisimilarity (Theorem 4.5);

(3) to prevent Trojan horses given by migrating secret ambients unleashing inner ambients without permission.

Somehow, both co-actions $\overline{\mathtt{in}}\langle n, h\rangle$ and $\overline{\mathtt{out}}\langle n, h\rangle$ represent entry point for ambients coming from outside and inside n, respectively.

Our labelled transition system can be smoothly adapted to SA[2] by removing passwords. *The resulting bisimilarity is a sound proof techniques for reduction barbed congruence in SA.* However, completeness results for SA, similar to Theorem 4.11, are very difficult to prove. The technical problem is due to the difficulty in conceiving a distinguishing context for actions like $\overline{\mathtt{enter}}\langle n\rangle$. Intuitively, in order to test that a process can allow entry to an ambient n a context has to move some 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 with *guarded choice* à la CCS. We believe that *in SA with guarded choice bisimilarity coincides with reduction barbed congruence.* The proof that bisimilarity implies reduction barbed congruence is basically the same as here. The interesting part is the converse where guarded choice plays a crucial role by allowing simple distinguishing contexts. As seen in Theorem 4.11 the only case where passwords are essential is when dealing with the action $\overline{\mathtt{enter}}\langle n\rangle$. In the presence of summation an easy distinguishing context for $\overline{\mathtt{enter}}\langle n\rangle$ is

$$[\cdot] \mid m[\mathtt{in}\langle n\rangle.\mathsf{SPY}_\alpha\langle n, h_1, h_2, R\rangle + \mathtt{in}\langle a\rangle] \mid a[\overline{\mathtt{in}}\langle a\rangle.\mathit{wrong}[\mathtt{out}\langle a\rangle]]$$

where names a and *wrong* are fresh. However, a general implementation of guarded choice is problematic as it involves non-local consensus decisions. For this reason we prefer our calculus SAP, which we believe is a good basis for developing interesting typing disciplines for mobile code using passwords, along the lines of [Bugliesi et al. 2005]. Even more, we think we can derive a labelled characterisation of typed barbed congruence along the lines of [Hennessy et al. 2003].

The type systems of Levi and Sangiorgi [Levi and Sangiorgi 2000] to ensure *immobility* and *single-threadness* can be easily adapted to SAP. However, due to the different of semantics of $\overline{\mathtt{out}}$, , the notion of single-threadness à la Levi and Sangiorgi does not necessarily guarantee the absence of *grave interferences.* For instance, according to Levi and Sangiorgi, the SAP system

$$c[\,b[\,a[\mathtt{out}\langle b\rangle] \mid \mathtt{in}\langle k\rangle\,] \mid \overline{\mathtt{out}}\langle b\rangle \mid k[\overline{\mathtt{in}}\langle k\rangle]\,]$$

should be considered single-thread, as every ambient has at most one firing capability. However, this system contains what can be considered a form of grave

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

interference, because the ambient b is shared by two different redexes which can give rise to logically different interactions. The interference is due to the ambient b which can engage two logically different interactions at the same time. Indeed, either a will exit b, ending up in c, or b will move into k, with a inside. So, a type system for single-threadness in SAP should be able to rule out similar situations. As a first attempt, one could require that any ambient should only contain (i) either an arbitrary number of parallel sub-ambients, possibly restricted; (ii) or a prefix $\alpha.P$, for some α and P. However, a type system of this kind would reject too many single-thread processes. Thus, the definition of an appropriate type system capturing single-threadness in SAP seems to be quite difficult.

The lts described in the current paper is a late version of that presented in the extended abstract [Merro and Hennessy 2002], where arbitrary processes provided by the environment were included in the lts as part of the label, and the definition of bisimilarity was standard. The two formulations are equivalent. However, a late lts allows the definition of both early and late bisimilarity. As already shown in [Merro and Zappa Nardelli 2003] for MA, we can prove early and late bisimilarity coincide also in our setting.

More recently, Jensen and Milner [Jensen and Milner 2004], based on previous work by Leifer and Milner [Leifer and Milner 2000], developed a meta-theory for distributed calculi to derive a minimal lts in such a way that the induced labelled bisimilarity is a congruence by construction. We conjecture that the application of those techniques to SAP will results in our early lts of [Merro and Hennessy 2002].

Taking inspiration from the current paper the first author, together with Francesco Zappa Nardelli, developed a semantic theory for MA [Merro and Zappa Nardelli 2003; 2004]. The main differences with respect to [Merro and Zappa Nardelli 2003; 2004] are the following:

—SAP differs from MA in that the former has co-capabilities and passwords, both features are essential to prove the characterisation result in SAP.

—In MA, unlike SAP, ambient mobility is asynchronous; no permission is required to migrate into an ambient. As noticed in [Sangiorgi 2001], this may cause a *stuttering* phenomenon in MA originated by ambients that may repeatedly enter and exit another ambient. As stuttering cannot be observed in MA, any successful characterisation of reduction barbed congruence in MA should not observe stuttering [Sangiorgi 2001].

—As co-capabilities allow the observation of the movements of a private ambient, the perfect firewall equation of MA does not hold in SAP, nor in SA. As a consequence, any bisimilarity in MA that wants to capture this law must not observe those movements.

—Higher-order actions in MA, unlike those in SAP, report the name of the migrating ambient. For instance, in MA the action $k.\mathtt{enter\_}n$ says that ambient k enters n. In MA, the knowledge of k is necessary to make the action observable for the environment. This is not needed in SAP, because movements can be observed by means of co-capabilities.

Apart from [Merro and Zappa Nardelli 2003; 2004], the bisimulation theory of the current paper has inspired several labelled bisimilarity for different distributed calculi such as Distributed π-calculus [Hennessy and Riely 1998], Seal [Vitek and Castagna 1999], a Calculus for Mobile Resources [Godskesen et al. 2002], NBA [Bugliesi et al. 2005], SafeDpi [Hennessy et al. 2003], and the Kell calculus [Schmitt and Stefani 2004]. The corresponding bisimilarities can be found in [Hennessy et al. 2003; Castagna and Zappa Nardelli 2002; Godskesen et al. 2002; Bugliesi et al. 2005; Hennessy et al. 2003; Schmitt and Stefani 2004], respectively, although only [Hennessy et al. 2003; Godskesen et al. 2002; Bugliesi et al. 2005; Hennessy et al. 2003; Schmitt and Stefani 2004] prove a characterisation result for a contextually defined notion of equivalence.

Unyapoth and Sewell [Unyapoth and Sewell 2001] takes a different, more intensional approach to define an equivalence for Nomadic Pict in that, in order to establish correctness of a particular protocol, a novel notion of equivalence based on coupled simulation tailored to accommodate code migration is identified. Although having many interesting properties, such as being a congruence, this equivalence is not shown to coincide with any independent contextually defined notion of equivalence.

Finally, more recently, Safe Ambients have been equipped with an equation based maximum sound theory à la Honda and Yoshida [Vigliotti and Phillips 2002].

ACKNOWLEDGMENT

We thank Julian Rathke, Davide Sangiorgi, and Francesco Zappa Nardelli for insightful discussions on higher-order process calculi and ambient-calculi. We also thank the anonymous referees for their very valuable remarks.

REFERENCES

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

Bugliesi, M., Castagna, G., and Crafa, S. 2001. Boxed ambients. In *Proc. 4th TACS*. LNCS, vol. 2215. Springer Verlag.

Bugliesi, M., Crafa, S., Merro, M., and Sassone, V. 2005. Communication interference in mobile boxed ambients. To appear in the journal of Information and Computation. An extended abstract appeared in Proc. FSTTCS'02, LNCS, Springer Verlag.

Cardelli, L. and Gordon, A. 1996. A commitment relation for the ambient calculus. Unpublished notes.

Cardelli, L. and Gordon, A. 2000. Mobile ambients. *Theoretical Computer Science 240,* 1, 177–213. An extended abstract appeared in *Proc. of FoSSaCS '98.*

Cardelli, L. and Gordon, A. D. 1999. Types for mobile ambients. In *Proc. of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM, 79–92.

Castagna, G. and Zappa Nardelli, F. 2002. The seal calculus revisited: Contextual equivalence and bisimilarity. In *Proc. 22nd FSTTCS '02*. LNCS, vol. 2556. Springer Verlag.

Ferrari, G., Montanari, U., and Tuosto, E. 2001. A LTS semantics of ambients via graph synchronization with mobility. In *Proc. ICTCS*. LNCS, vol. 2202. Springer Verlag.

Godskesen, J., Hildebrandt, T., and Sassone, V. 2002. A calculus of mobile resources. In *Proc. 10th CONCUR '02*. LNCS, vol. 2421. Springer Verlag.

Gordon, A. D. and Cardelli, L. 2002. Equational properties of mobile ambients. *Journal of Mathematical Structures in Computer Science 12*, 1–38. An extended abstract appeared in *Proc. FoSSaCs '99.*

Hennessy, M., Merro, M., and Rathke, J. 2003. Towards a behavioural theory of access and mobility control in distributed system. In *Proc. 5th FoSSaCS '03*. LNCS. Springer Verlag.

Hennessy, M., Rathke, J., and Yoshida, N. 2003. Safedpi: A language for controlling mobile code. Computer Science Report 2003:02, University of Sussex. An extended abstract appeared in the Proc. FOSSACS'04, volume 2987, Lecture Notes in Computer Science. Springer-Verlag 2004.

Hennessy, M. and Riely, J. 1998. A typed language for distributed mobile processes. In *Proc. 25th POPL*. ACM Press.

Honda, K. and Yoshida, N. 1994. Replication in Concurrent Combinators. In *Proc. TACS'94*. LNCS, vol. 789. Springer Verlag.

Honda, K. and Yoshida, N. 1995. On reduction-based process semantics. *Theoretical Computer Science 152,* 2, 437–486.

Jensen, O. H. and Milner, R. 2004. Bigraphs and mobile processes (revised). Tech. Rep. 580, LFCS, Dept. of Comp. Sci., Edinburgh Univ. Feb. An extended abstract appeared in *Conference Record of 30th Symposium on Principles of Programming Languages*, ACM Press, 2003.

Leifer, J. J. and Milner, R. 2000. Deriving bisimulation congruences for reactive systems. In *CONCUR 2000 - Concurrency Theory, 11th International Conference, University Park, PA, USA, August 22–25, 2000, Proceedings.* LNCS, vol. 1877. Springer-Verlag, 243–258.

Levi, F. and Sangiorgi, D. 2000. Controlling interference in ambients. An extended abstract appeared in *Proc. 27th Symposium on Principles of Programming Languages*, ACM Press.

Merro, M. and Hennessy, M. 2002. Bisimulation congruences in safe ambients. In *Proc. 29th POPL '02*. ACM Press.

Merro, M. and Zappa Nardelli, F. 2003. Bisimulation proof methods for mobile ambients. In *Proc. ICALP*. LNCS, vol. 2719. Springer Verlag.

Merro, M. and Zappa Nardelli, F. 2004. Behavioural theory for mobile ambients. In *IFIP TCS*. Kluwer, 549–562.

Milner, R. 1989. *Communication and Concurrency.* Prentice Hall.

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

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

Milner, R. and Sangiorgi, D. 1992. Barbed bisimulation. In *Proc. 19th ICALP*. LNCS, vol. 623. Springer Verlag, 685–695.

Phillips, A. 2005. The channel ambient calculus: From process algebra to mobile code. Ph.D. thesis, Imperial College London.

Phillips, A., Yoshida, N., and Eisenbach, S. 2004. A distributed abstract machine for boxed ambient calculi. In *Proc. ESOP*. LNCS, vol. 2987. Springer Verlag.

Sangiorgi, D. 1992. Expressing mobility in process algebras: First-order and higher-order paradigms. Ph.D. thesis, Department of Computer Science, University of Edinburgh.

Sangiorgi, D. 1994. The lazy lambda calculus in a concurrency scenario. *Information and Computation 111,* 1, 120–153.

Sangiorgi, D. 1996. Bisimulation for Higher-Order Process Calculi. *Information and Computation 131,* 2, 141–178.

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

Sangiorgi, D. and Milner, R. 1992. The problem of "Weak Bisimulation up to". In *Proc. CONCUR '92*. LNCS, vol. 630. Springer Verlag, 32–46.

Sangiorgi, D. and Walker, D. 2001a. *The π-calculus: a Theory of Mobile Processes.* Cambridge University Press.

Sangiorgi, D. and Walker, D. 2001b. Some results on barbed equivalences in pi-calculus. In *Proc. CONCUR '01*. LNCS, vol. 2154. Springer Verlag.

Schmitt, A. and Stefani, J. 2004. The kell calculus: A family of higher-order distributed process calculi. In *LNCS*. Springer-Verlag. Workshop of Global Computing 2004.

Unyapoth, A. and Sewell, P. 2001. Nomadic Pict: Correct communication infrastructures for mobile computation. In *Proc. 28th POPL*. ACM Press.

VIGLIOTTI, M. G. September 1999. Transition systems for the ambient calculus. Master thesis, Imperial College of Science, Technology and Medicine (University of London).

VIGLIOTTI, M. G. AND PHILLIPS, A. 2002. Barbs and congruences for safe mobile ambients. In *Electronic Notes in Theoretical Computer Science*. Vol. 66. Elsevier.

VITEK, J. AND CASTAGNA, G. 1999. Seal: A framework for secure mobile computations. In *Internet Programming Languages*. Number 1686 in LNCS. Springer Verlag, 47–77.

Received October 16 2003; July 27 2004; Accepted February 20 2005