Bisimulation Proof Methods for Mobile Ambients

Massimo Merro
Università di Verona, Italy

Francesco Zappa Nardelli
LIENS, Paris, France

Abstract

We study the behavioural theory of Cardelli and Gordon's Mobile Ambients. We give an *LTS* based operational semantics, and a labelled *bisimulation* based equivalence that coincides with reduction barbed congruence. We also provide two *up-to proof techniques* that we use to prove a set of *algebraic laws*, including the perfect firewall equation.

Keywords: process calculi, behavioural theories, bisimulation, Mobile Ambients, up-to proof techniques.

A mildly revised version of this paper appears in *Proc. ICALP 2003*, LNCS, Springer-Verlag, 2003. The full paper is available as Computer Science Report CS:2003:01, COGS, U. Sussex.

Address:

Massimo Merro: `merro@sci.unvr.it`
Dipartimento di Informatica, Ca' Vignal 2, Strada le Grazie 15, 37134 Verona, Italy
Fax + 39 045 80 27 928

Francesco Zappa Nardelli: `zappa@ens.fr`
LIENS (CNRS), Département d'Informatique, ENS, 45, rue d'Ulm, 75013 Paris, France
Fax: +33 1 4432 2156

Introduction

The calculus of *Mobile Ambients* [7], abbreviated MA, has been introduced as a process calculus for describing *mobile agents*.

In MA, the term $n[P]$ represents an agent, or *ambient*, named n, executing the code P. The ambient n is a bounded, protected, and (potentially) mobile space in which the computation P takes place. In turn P may contain other ambients, may perform (local) *communications*, or may exercise *capabilities*, which allow entry to or exit from named ambients. *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, [22], using the construct $(\boldsymbol{\nu} n)P$. A *system* in MA consists of a collection of ambients running in parallel where the knowledge of certain ambient names may be restricted.

Behavioural equality is a central idea in process calculi. For instance, behavioural equalities are used to verify that an implementation respects its specification, or that a code optimisation is correct. Roughly speaking, two processes are behavioural equivalent if no difference can be detected by interacting with them. In this paper, we focus on *bisimulation-based* behavioural equivalences. The notion of bisimulation was originally proposed in [24] and since then it has been used to define semantic equivalences for a variety of process calculi such as CCS, [21], and the π-calculus, [22].

Our touchstone equivalence is a generalisation of *reduction barbed congruence* of [16]. Reduction barbed congruence is the largest equivalence relation that (i) is a congruence, that is, it is preserved by the constructs of the language; (ii) preserves, in some sense, the reduction semantics of the language; (iii) preserves *barbs*, some simple observational property of terms.

However, context-based behavioural equalities, such as reduction barbed congruence, suffer from the universal quantification on contexts. Simpler proof techniques are based on *labelled bisimilarities* whose definitions do not use context quantification. These bisimilarities should imply, or (better) coincide with, reduction barbed congruence [25, 1, 11]. 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

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

Intuitively the action α in the judgement $P \xrightarrow{\boldsymbol{\alpha}} Q$ represents some small context P can interact with; if the labelled bisimilarity coincides with reduction barbed congruence then this collection of small contexts, codified as actions, is sufficient to capture all possible interactions that processes can have with arbitrary contexts.

Although the idea of bisimulation is very general and does not rely on the specific syntax of the calculus, the definition of an appropriate notion of bisimilarity for Mobile Ambients revealed harder than expected. The reasons of that can be resumed as follows:

- It is difficult for an ambient n to control interferences that may originate either from other ambients in its environment or from the computation running at n itself, [17].
- Ambient mobility is asynchronous — no permission is required to migrate into an ambient. As noticed in [29], this may cause a *stuttering* phenomenon originated by ambients that may repeatedly enter and exit another ambient. As stuttering cannot be observed, any successful characterisation of reduction barbed congruence should not observe stuttering as well [29].
- One of the main algebraic laws of MA is the *perfect firewall equation*, [7]:

  $$(\boldsymbol{\nu} n)n[P] = \mathbf{0} \qquad \text{for } n \text{ not in } P.$$

  If you suppose $P = \texttt{in_}k.\mathbf{0}$, it is evident that a bisimilarity that wants to capture this law must not observe the movements of *secret ambients*, that is those ambients, like n, whose names are not known by the rest of the system.

In [18], it is introduced a labelled bisimilarity for an "easier" variant of MA, called SAP, equipped with (i) *synchronous mobility*, as in Levi and Sangiorgi's *Safe Ambients* [17], and (ii) *passwords* to exercise control over, and differentiate between, different ambients that may wish to exercise a capability. The main result in [18] is the characterisation of reduction barbed congruence in terms of the labelled bisimilarity. The result holds only in SAP and crucially relies on the two features (i) and (ii) mentioned above.

The current paper is the natural continuation of [18] where, now, we tackle the original problem: *to provide bisimulation proof methods for Mobile Ambients.*

Contribution First of all, as in the Distributed π-calculus [14], we rewrite the syntax of MA in two levels: *processes* and *systems*. This is because we are interested in studying systems rather than processes. So, our behavioural equalities are defined over systems. This little expedient allows us to (i) focus on higher-order actions, where movement of code is involved, and (ii) to model stuttering in terms of standard τ-actions.

We introduce a new labelled transition system for MA which is used to define a labelled bisimilarity over systems. The resulting bisimilarity can be defined either in *late* or in *early* style. However, as in HOπ [26], the two formulations coincide, and we concentrate on the easier late version, denoted by $\approx$. The definition of $\approx$ reminds us that one of the asynchronous bisimilarity found in [1]. Indeed, as for inputs in asynchronous π, our bisimilarity does not observe the movements of secret ambients.

We prove that in MA the relation $\approx$ completely characterises reduction barbed congruence over systems. Then, we enhance our proof methods by defining two *up-to proof techniques*, along the lines of [23, 28, 31]. More precisely, we develop both *up-to expansion* and *up-to context* proof techniques and prove their soundness. We are not aware of other forms of up-to proof techniques for higher-order calculi. Finally, we apply our bisimulation proof methods to prove a collection of *algebraic laws* (among which the perfect firewall equation); then we also prove the correctness of the protocol, introduced in [7], for controlling access through a firewall.

The paper ends with a comparison with related work. For lack of space proofs are sketched or omitted; full proofs can be found in the full paper [19] available at `http://www.cogs.susx.ac.uk/reports.html`.

1 Mobile Ambients in Two Levels

In Table 1 we report the syntax of MA, where $\mathbf{N}$ denotes an infinite set of names. Unlike other definitions of MA in the literature, our syntax is defined in a two-level structure, a lower one for *processes*, and an upper one for *systems*.

The syntax for processes is standard [7] except for replication which is replaced by replicated prefixing, $!C.P$. As in the π-calculus replicated prefixing allows us to derive a simpler LTS; however, theory and results in this paper could be easily adapted for a calculus with full replication. For the sake of simplicity we omit *communication*; it will be added in Section 5. A system is a collection of ambients running in parallel where the knowledge of certain ambient names may be restricted among two or more ambients. We use a number of notational conventions. Parallel composition has the lowest precedence among the operators. 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 $n[\mathbf{0}]$. Restriction $(\boldsymbol{\nu} n)P$ acts as binder for name n, and the set of *free names* of P, $\mathrm{fn}(P)$, is defined accordingly.

A *static context* is a process context where the hole does not appear under a prefix or a replication. The dynamics of the calculus is specified by a *reduction relation*, $\twoheadrightarrow$, which is the least relation over processes closed under static contexts and satisfying the rules in Table 1. As systems are processes with a special structure, the rules of Table 1 also describe the evolution of systems. The *reduction semantics* relies on an auxiliary relation called *structural congruence* that brings the participants of a potential interaction into contiguous positions. We refer to [7] for the definition of structural congruence, $\equiv$. It is easy to check that the reduction relation is closed under systems, that is, systems always reduce to systems.

One of the main motivation of our work is the definition of a notion of labelled bisimilarity for MA. Rather than simply defining an ad-hoc bisimulation based equivalence over systems, we first introduce a basic equivalence by considering natural, desirable, properties.We choose to start from a generalisation of the *reduction barbed congruence*, [16], a *contextual*, *reduction closed*, and *barb preserving* equivalence relation. We now explain what these properties mean.

A *system context* is a context generated by the following grammar:

$$C[-] \quad ::= \quad - \quad \mid \quad C[-] \mid M \quad \mid \quad (\boldsymbol{\nu} n)C[-] \quad \mid \quad n[C[-] \mid P]$$

where M is an arbitrary system, and P is an arbitrary process. A relation $\mathcal{R}$ over systems is *contextual* if $M \,\mathcal{R}\, N$ implies $C[M] \,\mathcal{R}\, C[N]$ for all system contexts $C[-]$.

Table 1 The Mobile Ambients in Two Levels: Syntax and Reduction Rules

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

Systems:

M, N	::=	$\mathbf{0}$	termination
	$\mid$	$M_1 \mid M_2$	parallel composition
	$\mid$	$(\boldsymbol{\nu} n)M$	restriction
	$\mid$	$n[P]$	ambient

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

Capabilities:

C	::=	$\mathtt{in_}n$	may enter into n
	$\mid$	$\mathtt{out_}n$	may exit out of n
	$\mid$	$\mathtt{open_}n$	may open n

$n[\mathtt{in_}m.P \mid Q] \mid m[R] \rightarrow m[\, n[P \mid Q] \mid R\,]$	(Red In)
$m[\, n[\mathtt{out_}m.P \mid Q] \mid R\,] \rightarrow n[P \mid Q] \mid m[R]$	(Red Out)
$\mathtt{open_}n.P \mid n[\, Q\,] \rightarrow P \mid Q$	(Red Open)
$P \equiv Q \quad Q \rightarrow R \quad R \equiv S$ implies $P \rightarrow S$	(Red Struct)

A relation $\mathcal{R}$ over systems is *reduction closed* if $M \mathcal{R} N$ and $M \rightarrow M'$ implies the existence of some N' such that $N \rightarrow^* N'$ and $M' \mathcal{R} N'$, where $\rightarrow^*$ denotes the reflexive and transitive closure of $\rightarrow$.

In Mobile Ambients the observation predicate $M \downarrow_n$ denotes the possibility of the system M of interacting with the environment via the ambient n. We write $M \downarrow_n$ if $M \equiv (\boldsymbol{\nu}\tilde{m})(n[P] \mid M')$ with $n \notin \{\tilde{m}\}$. We write $M \Downarrow_n$ if there exists M' such that $M \rightarrow^* M'$ and $M' \downarrow_n$. A relation $\mathcal{R}$ over systems is *barb preserving* if $M \mathcal{R} N$ and $M \downarrow_n$ implies $N \Downarrow_n$.

Definition 1.1 (Reduction barbed congruence) *Reduction barbed congruence, written $\cong$, is the largest symmetric relation over systems which is reduction closed, contextual, and barb preserving.*

2 A Labelled Transition Semantics

In our language, the prefixes C give rise, in the standard manner, [21], to transitions of the form $P \xrightarrow{C} Q$; for example we have

$$\mathtt{in_}n.P_1 \mid P_2 \xrightarrow{\mathtt{in_}n} P_1 \mid P_2.$$

However, similarly to [18], each of the capability C induces different and more complicated actions. The LTS is defined over processes, although in the labelled bisimilarity we only consider actions going from systems to systems. We make a distinction between *pre-actions* and *env-actions*: the former denote the possibility to exercise certain capabilities whereas the latter model the interaction of a system with its environment. As usual, we also have *τ-actions* to model internal computations. Only env-actions and τ-actions model the evolution of a system at run-time.

The pre-actions, defined in Table 3, are of the form $P \xrightarrow{\boldsymbol{\pi}} O$ where the ranges of π and of O, the *outcomes*, are reported in Table 2. An outcome may be a simple process Q, if for example π is a prefix of the language, or a *concretion*, of the form $(\boldsymbol{\nu}\tilde{m})\langle P\rangle Q$, when an ambient boundary is somehow involved. In this case, intuitively, P represents the part of the system affected by the action, while Q is not affected, and $\tilde{m}$ is the set of private names shared by P and Q. We adopt the convention that if K is the concretion $(\boldsymbol{\nu}\tilde{m})\langle P\rangle Q$, then $(\boldsymbol{\nu} r)K$ is a shorthand for $(\boldsymbol{\nu}\tilde{m})\langle P\rangle(\boldsymbol{\nu} r)Q$, if $r \notin \mathrm{fn}(P)$, and the concretion $(\boldsymbol{\nu} r\tilde{m})\langle P\rangle Q$ otherwise. We

have a similar convention for the rule (π Par): $K \mid R$ is defined to be the concretion $(\boldsymbol{\nu}\tilde{m})\langle P\rangle(Q \mid R)$, where $\tilde{m}$ are chosen, using α-conversion if necessary, so that $\mathrm{fn}(R)\cap\{\tilde{m}\} = \emptyset$. Occasionally, we omit dead processes when they are in parallel with processes, writing P for $P \mid \mathbf{0}$.

The τ-actions, defined in Table 4, model the internal evolution of processes. The env-actions, defined in Table 5, are of the form $M \xrightarrow{\boldsymbol{\sigma}} M'$, where the range of σ is given in Table 2. In practice, env-actions turn concretions into running systems by explicitly introducing the environment's ambient interacting with the process in question. The content of this ambient will be instantiated later, in the bisimilarity, with a process. For convenience, we extend the syntax of processes with the special process $\circ$ to pinpoint those ambients whose content will be instantiated later. The process $\circ$ does not reduce: it is simply a placeholder. Notice that, unlike pre-actions and τ-actions, env-actions do not have structural rules; this is because env-actions are supposed to be performed by systems that can directly interact with the environment.

We call *actions* the set of env-actions extended with τ. As our bisimilarity will be defined over systems, we will only consider actions (and not pre-actions) in its definition.

Proposition 2.1 *If T is a system (resp. a process), and $T \xrightarrow{\boldsymbol{\alpha}} T'$, then T' is a system (resp. a process), possibly containing the special process $\circ$.*

Now, we explain the rules induced by the the prefix $\mathtt{in}$, the *immigration* of ambients. A typical example of an ambient m migrating into an ambient n follows:

$$(\boldsymbol{\nu}m)(m[\,\mathtt{in}_n.P_1 \mid P_2\,] \mid M) \mid n[Q] \twoheadrightarrow (\boldsymbol{\nu}m)(M \mid n[\,m[\,P_1 \mid P_2\,] \mid Q])$$

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

$$m[\mathtt{in}_n.P_1 \mid P_2] \xrightarrow{\mathtt{enter}_n} \langle m[P_1 \mid P_2]\rangle\mathbf{0}$$

and more generally, using the structural rules (π Res) and (π Par),

$$(\boldsymbol{\nu}m)(m[\mathtt{in}_n.P_1 \mid P_2] \mid M) \xrightarrow{\mathtt{enter}_n} (\boldsymbol{\nu}m)\langle m[P_1 \mid P_2]\rangle M.$$

This means that the ambient $m[\mathtt{in}_n.P_1 \mid P_2]$ has the capability to enter an ambient n; if the capability is exercised, the ambient $m[P_1 \mid P_2]$ will enter n while M will be the residual at the point of execution. Of course the action can only be executed if there is an ambient n in parallel. The rule (π Amb) allows to check for the presence of ambients. So for example, we have

$$n[Q] \xrightarrow{\mathtt{amb}_n} \langle Q\rangle\mathbf{0}.$$

Here, the concretion $\langle Q\rangle\mathbf{0}$ says that Q is in n, while $\mathbf{0}$ is outside. Finally, the communication (τ Enter) allows these two complementary actions to occur simultaneously, executing the migration of the ambient $m[P_1 \mid P_2]$ from its current computation space into the ambient n, giving rise to the original move above:

$$(\boldsymbol{\nu}m)(m[\,\mathtt{in}_n.P_1 \mid P_2\,] \mid M) \mid n[Q] \xrightarrow{\boldsymbol{\tau}} (\boldsymbol{\nu}m)(M \mid n[\,m[\,P_1 \mid P_2\,] \mid Q]).$$

Note that this is a *higher-order* interaction, as the ambient $m[P_1 \mid P_2]$ is transferred between two computation spaces.

We have not said yet what env-actions are useful for. They model the interaction of mobile agents with their environment. So, for instance, using the rule (Enter Shh), we derive from

$$(\boldsymbol{\nu}m)(m[\mathtt{in}_n.P_1 \mid P_2] \mid M) \xrightarrow{\mathtt{enter}_n} (\boldsymbol{\nu}m)\langle m[P_1 \mid P_2]\rangle M.$$

the transition

$$(\boldsymbol{\nu}m)(m[\mathtt{in}_n.P_1 \mid P_2] \mid M) \xrightarrow{*.\mathtt{enter}_n} (\boldsymbol{\nu}m)(n[\circ \mid m[P_1 \mid P_2]] \mid M).$$

This transition denotes a private (*secret*) ambient entering an ambient n provided by the environment. The computation running at n can be added later by instantiating the placeholder $\circ$.

Table 2 Pre-actions, Env-actions, Actions, Concretions, and Outcomes

$$
\begin{array}{llcl}
\textit{Pre-actions:} & \pi & ::= & \mathtt{in}_n \mid \mathtt{out}_n \mid \mathtt{open}_n \mid \mathtt{enter}_n \mid \mathtt{amb}_n \mid \mathtt{exit}_n \\
\textit{Outcomes:} & O & ::= & P \mid K \\
\textit{Actions:} & \alpha & ::= & \sigma \cup \tau
\end{array}
\qquad
\begin{array}{llcl}
\textit{Env-actions:} & \sigma & ::= & k.\mathtt{enter}_n \mid k.\mathtt{exit}_n \mid *.\mathtt{enter}_n \mid *.\mathtt{exit}_n \mid n.\overline{\mathtt{enter}}_k \mid k.\mathtt{open}_n \\
\textit{Concretions:} & K & ::= & (\boldsymbol{\nu}\tilde{m})\langle P\rangle Q
\end{array}
$$

Table 3 Labelled Transition System - Pre-actions

$$(\pi \text{ Pfx})\ \frac{-}{\pi.P \xrightarrow{\pi} P} \qquad (\pi \text{ Repl Pfx})\ \frac{-}{!\pi.P \xrightarrow{\pi} P \mid !\pi.P} \qquad (\pi \text{ Exit})\ \frac{P \xrightarrow{\mathtt{out}_n} P_1}{m[P] \xrightarrow{\mathtt{exit}_n} \langle m[P_1]\rangle \mathbf{0}}$$

$$(\pi \text{ Amb})\ \frac{-}{n[P] \xrightarrow{\mathtt{amb}_n} \langle P\rangle \mathbf{0}} \qquad (\pi \text{ Enter})\ \frac{P \xrightarrow{\mathtt{in}_n} P_1}{m[P] \xrightarrow{\mathtt{enter}_n} \langle m[P_1]\rangle \mathbf{0}} \qquad (\pi \text{ Res})\ \frac{P \xrightarrow{\pi} O \qquad n \notin \mathrm{fn}(\pi)}{(\boldsymbol{\nu} n)P \xrightarrow{\pi} (\boldsymbol{\nu} n)O}$$

$$(\pi \text{ Par})\ \frac{P \xrightarrow{\pi} O}{\begin{array}{l} P \mid Q \xrightarrow{\pi} O \mid Q \\ Q \mid P \xrightarrow{\pi} Q \mid O \end{array}}$$

Table 4 Labelled Transition System - τ-actions

$$(\tau \text{ Enter})\ \frac{P \xrightarrow{\mathtt{enter}_n} (\boldsymbol{\nu}\tilde{p})\langle P_1\rangle P_2 \quad Q \xrightarrow{\mathtt{amb}_n} (\boldsymbol{\nu}\tilde{q})\langle Q_1\rangle Q_2 \ {}^{(*)}}{\begin{array}{l} P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p})(\boldsymbol{\nu}\tilde{q})(n[P_1 \mid Q_1] \mid P_2 \mid Q_2) \\ Q \mid P \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{q})(\boldsymbol{\nu}\tilde{p})(n[Q_1 \mid P_1] \mid Q_2 \mid P_2) \end{array}} \qquad (\tau \text{ Par})\ \frac{P \xrightarrow{\tau} P'}{\begin{array}{l} P \mid Q \xrightarrow{\tau} P' \mid Q \\ Q \mid P \xrightarrow{\tau} Q \mid P' \end{array}}$$

$$(\tau \text{ Exit})\ \frac{P \xrightarrow{\mathtt{exit}_n} (\boldsymbol{\nu}\tilde{m})\langle k[P_1]\rangle P_2}{n[P] \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{m})(k[P_1] \mid n[P_2])} \qquad (\tau \text{ Amb})\ \frac{P \xrightarrow{\tau} Q}{n[P] \xrightarrow{\tau} n[Q]}$$

$$(\tau \text{ Open})\ \frac{P \xrightarrow{\mathtt{open}_n} P_1 \quad Q \xrightarrow{\mathtt{amb}_n} (\boldsymbol{\nu}\tilde{m})\langle Q_1\rangle Q_2}{\begin{array}{l} P \mid Q \xrightarrow{\tau} P_1 \mid (\boldsymbol{\nu}\tilde{m})(Q_1 \mid Q_2) \\ Q \mid P \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{m})(Q_1 \mid Q_2) \mid P_1 \end{array}} \qquad (\tau \text{ Res})\ \frac{P \xrightarrow{\tau} P'}{(\boldsymbol{\nu} n)P \xrightarrow{\tau} (\boldsymbol{\nu} n)P'}$$

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

Table 5 Labelled Transition System - Env-actions

$$(\text{Enter})\ \frac{P \xrightarrow{\mathtt{enter}_n} (\boldsymbol{\nu}\tilde{m})\langle k[P_1]\rangle P_2 \ {}^{(\dagger)}}{P \xrightarrow{k.\mathtt{enter}_n} (\boldsymbol{\nu}\tilde{m})(n[\circ \mid k[P_1]] \mid P_2)} \qquad (\text{Exit})\ \frac{P \xrightarrow{\mathtt{exit}_n} (\boldsymbol{\nu}\tilde{m})\langle k[P_1]\rangle P_2 \ {}^{(\dagger)}}{P \xrightarrow{k.\mathtt{exit}_n} (\boldsymbol{\nu}\tilde{m})(k[P_1] \mid n[\circ \mid P_2])}$$

$$(\text{Co-Enter})\ \frac{P \xrightarrow{\mathtt{amb}_n} (\boldsymbol{\nu}\tilde{m})\langle P_1\rangle P_2 \ {}^{(\dagger)}}{P \xrightarrow{n.\overline{\mathtt{enter}}_k} (\boldsymbol{\nu}\tilde{m})(n[P_1 \mid k[\circ]] \mid P_2)} \qquad (\text{Open})\ \frac{P \xrightarrow{\mathtt{amb}_n} (\boldsymbol{\nu}\tilde{m})\langle P_1\rangle P_2}{P \xrightarrow{k.\mathtt{open}_n} k[\circ \mid (\boldsymbol{\nu}\tilde{m})(P_1 \mid P_2)]}$$

$$(\text{Enter Shh})\ \frac{P \xrightarrow{\mathtt{enter}_n} (\boldsymbol{\nu}\tilde{m})\langle k[P_1]\rangle P_2 \ {}^{(\ddagger)}}{P \xrightarrow{*.\mathtt{enter}_n} (\boldsymbol{\nu}\tilde{m})(n[\circ \mid k[P_1]] \mid P_2)} \qquad (\text{Exit Shh})\ \frac{P \xrightarrow{\mathtt{exit}_n} (\boldsymbol{\nu}\tilde{m})\langle k[P_1]\rangle P_2 \ {}^{(\ddagger)}}{P \xrightarrow{*.\mathtt{exit}_n} (\boldsymbol{\nu}\tilde{m})(k[P_1] \mid n[\circ \mid P_2])}$$

(†) We require $k \notin \tilde{m}$. (‡) We require $k \neq n$ and $k \in \tilde{m}$

Had the ambient name m not been restricted, we would have used the rule (Enter) to derive

$$m[\mathtt{in}_n.P_1 \mid P_2] \mid M \xrightarrow{\boldsymbol{m.\mathtt{enter}_n}} n[\circ \mid m[P_1 \mid P_2]] \mid M$$

to model a global ambient m that enters an ambient n provided by the environment.

The rules for *emigration* and *opening* follow the same lines.

Whenever a system offers a public ambient n at top-level, a context can interact with the system by providing an ambient entering n. The rule Co-enter captures this interaction between system and environment.

We end this section with a theorem that asserts that the LTS based semantics coincides with the reduction semantics of Section 1

Theorem 2.2 *If $P \xrightarrow{\boldsymbol{\tau}} P'$ then $P \rightarrow P'$. If $P \rightarrow P'$ then $P \xrightarrow{\boldsymbol{\tau}} \equiv P'$.*

From the result above, it is easy to establish that if $M \cong N$ then (i) $M \Downarrow n$ iff $N \Downarrow n$ and (ii) $M \Rightarrow M'$ implies there is N' such that $N \Rightarrow N'$ and $M' \cong N'$.

3 Characterising Reduction Barbed Congruence

In this section we define a labelled bisimilarity for MA that completely characterises reduction barbed congruence.

Since we are interested in *weak bisimilarities*, that abstract over τ-actions, we introduce the notion of weak action. The definition is standard: $\Rightarrow$ denotes the reflexive and transitive closure of $\xrightarrow{\boldsymbol{\tau}}$; $\stackrel{\boldsymbol{\alpha}}{\Longrightarrow}$ denotes $\Rightarrow \xrightarrow{\boldsymbol{\alpha}} \Rightarrow$; $\stackrel{\boldsymbol{\hat{\alpha}}}{\Longrightarrow}$ denotes $\Rightarrow$ if $\alpha = \tau$ and $\stackrel{\boldsymbol{\alpha}}{\Longrightarrow}$ otherwise.

In the previous section we said that actions (and more precisely env-actions) introduce a special process $\circ$ to pinpoint those ambients whose content will be specified in the bisimilarity. The $\bullet$ operator instatiates the placeholder with a process, as defined below. It performs a name-capture avoiding substitution.

Definition 3.1 *Let T, T_1, and T_2 range over both systems and processes. Then, for a process P, we define:*

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl}
\mathbf{0} \bullet P & \stackrel{\text{def}}{=} & \mathbf{0} & (T_1 \mid T_2) \bullet P & \stackrel{\text{def}}{=} & (T_1 \bullet P) \mid (T_2 \bullet P) & \circ \bullet P & \stackrel{\text{def}}{=} & P \\
n[R] \bullet P & \stackrel{\text{def}}{=} & n[R \bullet P] & (\boldsymbol{\nu} n)T \bullet P & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} n)(T \bullet P) \text{ if } n \notin \mathrm{fn}(P) & & & \\
!C.R \bullet P & \stackrel{\text{def}}{=} & !C.(R \bullet P) & C.R \bullet P & \stackrel{\text{def}}{=} & C.(R \bullet P.) & & &
\end{array}$$

Everything is in place to define our bisimilarity.

Definition 3.2 (Late bisimilarity) *A symmetric relation $\mathcal{R}$ over systems is a* late bisimulation *if $M \mathcal{R} N$ implies:*

- *if $M \xrightarrow{\boldsymbol{\alpha}} M'$, $\alpha \notin \{*.\mathtt{enter}_n, *.\mathtt{exit}_n\}$, then there is a system N' such that $N \stackrel{\boldsymbol{\hat{\alpha}}}{\Longrightarrow} N'$ and for all processes P it holds $M' \bullet P \mathcal{R} N' \bullet P$;*
- *if $M \xrightarrow{\boldsymbol{*.\mathtt{enter}_n}} M'$ then there is a system N' such that $N \mid n[\circ] \Rightarrow N'$ and for all processes P it holds $M' \bullet P \mathcal{R} N' \bullet P$;*
- *if $M \xrightarrow{\boldsymbol{*.\mathtt{exit}_n}} M'$ then there is a system N' such that $n[\circ \mid N] \Rightarrow N'$ and for all processes P it holds $M' \bullet P \mathcal{R} N' \bullet P$.*

Systems M and N are late bisimilar*, written $M \approx N$, if $M \mathcal{R} N$ for some late bisimulation $\mathcal{R}$.*

The bisimilarity above has a universal quantification over the process P provided by the environment. This process instantiates the special process $\circ$ generated via env-actions. The bisimilarity is defined in a *late* style as the existential quantification precedes the universal one. Another possibility would be to define the bisimilarity in *early* style where the universal quantification over the environment's contribution P precedes that over the derivative N'. We write $\approx_{\mathrm{e}}$ to denote this early variant. By definition, every late bisimulation is also a early one, while the converse, in general, does not hold. However, in our case, as in HOπ [26], we

Table 6 Contexts for visible actions

$\alpha = k.\mathtt{enter}_n$	$C_\alpha[-] = n[\circ \mid \mathsf{done}[\mathtt{in}_k.\mathtt{out}_k.\mathtt{out}_n]] \mid -$
$\alpha = k.\mathtt{exit}_n$	$C_\alpha[-] = (\boldsymbol{\nu} a)a[\mathtt{in}_k.\mathtt{out}_k.\mathsf{done}[\mathtt{out}_a]] \mid n[\circ \mid -]$
$\alpha = n.\overline{\mathtt{enter}}_k$	$C_\alpha[-] = (\boldsymbol{\nu} a)a[\mathtt{in}_n.k[\mathtt{out}_a.(\circ \mid (\boldsymbol{\nu} b)b[\mathtt{out}_k.\mathtt{out}_n.\mathsf{done}[\mathtt{out}_b]])]] \mid -$
$\alpha = k.\mathtt{open}_n$	$C_\alpha[-] = k[\circ \mid (\boldsymbol{\nu} a, b)(\mathtt{open}_b.\mathtt{open}_a.\mathsf{done}[\mathtt{out}_k] \mid a[- \mid \mathtt{open}_n.b[\mathtt{out}_a]])]$ where a and b are fresh.

will prove that late and early bisimilarity actually coincide. As a consequence, late bisimilarity will be our main labelled bisimilarity because the derivatives N' do not depend on processes P.

Finally, notice that actions $*.\mathtt{enter}_n$ and $*.\mathtt{exit}_n$ are treated apart asking for weaker matching requirements. This is because both actions are not observable. Somehow, this is very similar to what happens with input actions in the asynchronous π-calculus [15, 3].

3.1 Soundness

Late and early bisimilarity represent two proof techniques for reduction barbed congruence. More precisely we prove that they are both contextual and contained in reduction barbed congruence.

Theorem 3.3 *Late bisimilarity is contextual.*

The above result does not rely on the transitivity of the late bisimulation. Moreover, the proof can be easily adapted to prove that also the early bisimilarity is contextual.

Proposition 3.4 *Late bisimilarity is an equivalence relation.*

Proof [Sketch] The only non-trivial property is transitivity. We basically need Theorem 3.3 to say that $\approx$ is preserved by parallel composition and ambient nesting. These two properties are necessary to deal with the env-actions $*.\mathtt{enter}_n$ and $*.\mathtt{exit}_n$. □

Proposition 3.5 *Early bisimilarity is contextual, and it is an equivalence relation.*

We conclude that both late and early bisimilarity are contained in the reduction barbed congruence.

Theorem 3.6 (Soundness) *The following inclusions hold* $\approx \subseteq \approx_{\mathrm{e}} \subseteq \cong$.

Proof The first inclusion holds by definition. The second one comes from Proposition 3.5 and the fact that early bisimilarity is reduction closed and barb-preserving. □

3.2 Completeness

We now prove that late and early bisimilarity are more than proof techniques. They actually characterise reduction barbed congruence. The main challenge here is to design the contexts capable to observe our visible actions. The definition of these contexts, $C_\alpha[-]$, for every visible action α, is given in Table 6. The special ambient name done is used as *fresh* barb to signal the consumption of actions.

To prove our characterisation result it suffices to show that reduction barbed congruence is contained in the late bisimilarity. Then, by Theorem 3.6 we can conclude that late, early, and reduction barbed congruence, they all coincide. The proof that reduction barbed congruence implies the late bisimilarity requires the correspondence between visible actions α and their corresponding contexts $C_\alpha[-]$.

The context for $n.\overline{\mathtt{enter}}_k$ deserves some explication. The secret ambient b assures that if the ambient done arrives at top-level, then it is empty. This allows a uniform formulation of Lemma 3.8. The need for the secret ambient a is more elusive. It prevents that the ambient k is used to enter into the ambient n by a Trojan horse hidden in the system plugged into the hole. This could invalidate Lemma 3.8.

The following lemma says that the defining contexts are sound, that is, they can successfully mimic the execution of visible actions.

Table 7 Auxiliary contexts and processes

$$
\begin{array}{lcl}
-_1 \oplus -_2 & = & (\boldsymbol{\nu} o)(o[\,] \mid \texttt{open_}o.-_1 \mid \texttt{open_}o.-_2) \\
\mathrm{SPY}_\alpha\langle i,j,-\rangle & = & (i[\texttt{out_}n] \mid -) \oplus (j[\texttt{out_}n] \mid -) \\
 & & \quad \text{if } \alpha \in \{k.\texttt{enter_}n, k.\texttt{exit_}n, k.\texttt{open_}n, *.\texttt{enter_}n, *.\texttt{exit_}n\} \\
\mathrm{SPY}_\alpha\langle i,j,-\rangle & = & (i[\texttt{out_}k.\texttt{out_}n] \mid -) \oplus (j[\texttt{out_}k.\texttt{out_}n] \mid -) \text{ if } \alpha \in \{n.\overline{\texttt{enter}}\texttt{_}k\}
\end{array}
$$

Lemma 3.7 *Let M be a system, and let $\alpha \in \{k.\texttt{enter_}n, k.\texttt{exit_}n, n.\overline{\texttt{enter}}\texttt{_}k, k.\texttt{open_}n\}$. For all processes P, if $M \xrightarrow{\alpha} M'$, then $C_\alpha[M] \bullet P \Rrightarrow \cong M' \bullet P \mid \mathit{done}[\,]$.*

To complete the correspondence proof between actions α and their contexts $C_\alpha[-]$, we have to prove the converse of Lemma 3.7, formalised in Lemma 3.8. Such result requires a few technical definitions given in Table 7. The symbol $\oplus$ denotes a form of internal choice, whereas the context $\mathrm{SPY}_\alpha\langle i,j,-\rangle$ is a technical tool to guarantee that the process P provided by the environment does not perform any action. This is necessary when proving the completeness to guarantee that the contribution of P is the same on both sides. The ability of $\mathrm{SPY}_\alpha\langle i,j,P\rangle$ to "spy" on P stems from the fact that one of the two fresh barbs i and j is lost when P performs any action.

Lemma 3.8 *Let M be a system, let $\alpha \in \{k.\texttt{enter_}n, k.\texttt{exit_}n, n.\overline{\texttt{enter}}\texttt{_}k, k.\texttt{open_}n\}$, and let i,j be fresh names for M. For all processes P with $\{i,j\} \cap \mathrm{fn}(P) = \emptyset$, if $C_\alpha[M] \bullet \mathrm{SPY}_\alpha\langle i,j,P\rangle \Rrightarrow \cong O \mid \mathit{done}[\,]$ and $O \Downarrow_{i,j}$, then there exists a system M' such that $O \cong M' \bullet \mathrm{SPY}_\alpha\langle i,j,P\rangle$ and $M \xRightarrow{\alpha} M'$.*

Proof [Sketch] The proof depends on the precise definition of the context. The main argument is that in the reduction

$$C_\alpha[M] \bullet \mathrm{SPY}_\alpha\langle i,j,P\rangle \Rrightarrow \cong O \mid \mathsf{done}[\,]$$

the fresh ambient $\mathsf{done}[\,]$ can only be unleashed if M performs the action α, possibly preceeded or followed by some internal actions. The fresh barbs i,j assure that the process P does not take part in the reduction, and that the component $\mathrm{SPY}_\alpha\langle i,j,P\rangle$ is found intact after the reduction. □

Theorem 3.9 (Completeness) *Reduction barbed congruence is contained in late bisimilarity.*

Proof [Sketch] We prove that the relation $\mathcal{R} = \{(M,N) \mid M \cong N\}$ is a late bisimulation. The result will then follow by co-induction.

Suppose $M \mathcal{R} N$ and $M \xrightarrow{\alpha} M'$. Suppose also $\alpha \in \{k.\texttt{enter_}n, k.\texttt{exit_}n, n.\overline{\texttt{enter}}\texttt{_}k, k.\texttt{open_}n\}$. We must find a system N' such that $N \xRightarrow{\alpha} N'$ and for all P, $M' \bullet P \cong N' \bullet P$.

The idea of the proof is to use a particular context which mimics the effect of the action α, and also allows us to subsequentely compare the residuals of the two systems. This context has the form

$$D_\alpha\langle P\rangle[-] \;=\; (C_\alpha[-] \mid \mathsf{Flip}) \bullet \mathrm{SPY}_\alpha\langle i,j,P\rangle$$

where $C_\alpha[-]$ are the contexts in Table 6 and Flip is the system:

$$(\boldsymbol{\nu} k)k[\texttt{in_done.out_done.}(\mathsf{succ}[\texttt{out_}k] \oplus \mathsf{fail}[\texttt{out_}k])]$$

with succ and fail fresh names. Intuitively, the existence of the fresh barb fail indicates that the action α has not yet happened, whereas the presence of succ together with the absence of fail ensures that the action α has been performed, and has been reported via done.

As $\cong$ is contextual, $M \cong N$ implies that, for all processes P, it holds $D_\alpha\langle P\rangle[M] \;\cong\; D_\alpha\langle P\rangle[N]$. By Lemma 3.7, and by inspecting the reduction of the Flip process, we observe that:

$$D_\alpha\langle P\rangle[M] \;\Rrightarrow\cong\; M' \bullet \mathrm{SPY}_\alpha\langle i,j,P\rangle \mid \mathsf{done}[\,] \mid \mathsf{Flip} \;\Rrightarrow\cong\; M' \bullet \mathrm{SPY}_\alpha\langle i,j,P\rangle \mid \mathsf{done}[\,] \mid \mathsf{succ}[\,]$$

where $M' \bullet \mathrm{SPY}_\alpha\langle i,j,P\rangle \mid \mathsf{done}[\,] \mid \mathsf{succ}[\,] \Downarrow_{i,j,\mathsf{succ}} \not\Downarrow_{\mathsf{fail}}$. Call this outcome O_1. This reduction must be matched by a corresponding reduction $D_\alpha\langle P\rangle[N] \Rrightarrow O_2$, where $O_1 \cong O_2$. However, the possible matching reductions are constrained by the barbs of O_1, because it must hold $O_2 \Downarrow_{i,j,\mathsf{succ}} \not\Downarrow_{\mathsf{fail}}$.

As $O_2 \Downarrow_{\mathsf{succ}} \not\Downarrow_{\mathsf{fail}}$, it must be $O_2 \cong \hat{N} \mid \mathsf{done}[\,] \mid \mathsf{succ}[\,]$, for some systems $\hat{N}$. As $O_2 \Downarrow_{i,j}$, the previous observation can be combined with Lemma 3.8 to derive the existence of a system (over the extended process syntax) N' such that $\hat{N} \cong N' \bullet \mathrm{SPY}_\alpha\langle i,j,P\rangle$ and a weak action

$$N \stackrel{\boldsymbol{\alpha}}{\Longrightarrow} N'.$$

To conclude we must establish that for all P, it holds $M' \bullet P \cong N' \bullet P$. As barbed congruence is preserved by restriction, we have $(\boldsymbol{\nu}\mathsf{done},\mathsf{succ})O_1 \cong (\boldsymbol{\nu}\mathsf{done},\mathsf{succ})O_2$. As $(\boldsymbol{\nu}\mathsf{done})\mathsf{done}[\,] \cong (\boldsymbol{\nu}\mathsf{succ})\mathsf{succ}[\,] \cong \mathbf{0}$, it follows that

$$M' \bullet \mathrm{SPY}_\alpha\langle i,j,P\rangle \;\cong\; N' \bullet \mathrm{SPY}_\alpha\langle i,j,P\rangle.$$

Again, $\cong$ is preserved by restriction and $(\boldsymbol{\nu} i,j)\mathrm{SPY}_\alpha\langle i,j,P\rangle \cong P$. So, we can finally derive $M' \bullet P \;\mathcal{R}\; N' \bullet P$, for all processes P.

To complete the proof we need to consider the actions $*.\mathtt{enter}_n$ and $*.\mathtt{exit}_n$: as they are not observable, these cases are much easier. □

Theorem 3.10 *Late bisimilarity, early bisimilarity, and reduction barbed congruence they all coincide.*

Proof Theorem 3.6 states that $\approx \subseteq \approx_{\mathrm{e}}$ and $\approx_{\mathrm{e}} \subseteq \cong$. Theorem 3.9 states the reduction barbed congruence is contained in late bisimilarity, that is $\cong \subseteq \approx$. We hence have the following chain of inclusions $\cong \subseteq \approx \subseteq \approx_{\mathrm{e}} \subseteq \cong$. □

4 Up-to Proof Techniques

In the previous section we gave a labelled characterisation of reduction barbed congruence to prove that two systems have the same behaviour. In this section we adapt some well-known *up-to* proof techniques [23, 28] to our setting. These techniques allow us to reduce the size of the relation $\mathcal{R}$ to exhibit to prove that two processes are bisimilar. We focus on the *up-to expansion* [30] and the *up-to context* technique [27]. As in π-calculus, these two techniques can be merged: for lack of space and we only report the resulting technique.

Roughly, the expansion [2], written $\lesssim$, is an asymmetric variant of the bisimilarity that allows us to count the number of silent moves performed by a process. More precisely, $M \lesssim N$ holds if M and N are bisimilar and N has at least as many τ-moves as M. Formally,

Definition 4.1 (Expansion) *A relation $\mathcal{R}$ over systems is an* expansion *if $M \,\mathcal{R}\, N$ implies:*

- *if $M \stackrel{\boldsymbol{\alpha}}{\longrightarrow} M'$, $\alpha \notin \{*.\mathtt{enter}_n, *.\mathtt{exit}_n\}$, then there exists a system N' such that $N \stackrel{\hat{\boldsymbol{\alpha}}}{\Longrightarrow} N'$ and for all processes P it holds $M' \bullet P \,\mathcal{R}\, N' \bullet P$;*
- *if $M \stackrel{*.\mathtt{enter}_\boldsymbol{n}}{\longrightarrow} M'$ then there is a system N' such that $N \mid n[\,\circ\,] \Rightarrow N'$ and for all processes P it holds $M' \bullet P \,\mathcal{R}\, N' \bullet P$;*
- *if $M \stackrel{*.\mathtt{exit}_\boldsymbol{n}}{\longrightarrow} M'$ then there is a system N' such that $n[\circ \mid N] \Rightarrow N'$ and for all processes P it holds $M' \bullet P \,\mathcal{R}\, N' \bullet P$;*
- *if $N \stackrel{\boldsymbol{\alpha}}{\longrightarrow} N'$, $\alpha \notin \{*.\mathtt{enter}_n, *.\mathtt{exit}_n\}$, then there exists a system M' such that $M \stackrel{\hat{\boldsymbol{\alpha}}}{\longrightarrow} M'$ and for all processes P it holds $M' \bullet P \,\mathcal{R}\, N' \bullet P$;*
- *if $N \stackrel{*.\mathtt{enter}_\boldsymbol{n}}{\longrightarrow} N'$ then $(M \mid n[P]) \,\mathcal{R}\, N' \bullet P$, for all processes P;*
- *if $N \stackrel{*.\mathtt{exit}_\boldsymbol{n}}{\longrightarrow} N'$ then $n[M \mid P] \,\mathcal{R}\, N' \bullet P$, for all processes P.*

We write $M \lesssim N$, if $M \,\mathcal{R}\, N$ for some expansion $\mathcal{R}$.

Definition 4.2 (Bisimulation up-to context and up-to $\gtrsim$) *A symmetric relation $\mathcal{R}$ is a* bisimulation up-to context and up-to $\gtrsim$ *if $P \,\mathcal{R}\, Q$ implies:*

- *if $M \stackrel{\boldsymbol{\alpha}}{\longrightarrow} M''$, $\alpha \notin \{*.\mathtt{enter}_n, *.\mathtt{exit}_n\}$, then there exists a system N'' such that $N \stackrel{\hat{\boldsymbol{\alpha}}}{\Longrightarrow} N''$, and for all processes P there is a system context $C[-]$ and systems M' and N' such that $M'' \bullet P \gtrsim C[M']$, $N'' \bullet P \gtrsim C[N']$, and $M' \,\mathcal{R}\, N'$;*

- *if* $M \xrightarrow{\texttt{*.enter_}n} M''$ *then there exists a system* N'' *such that* $N \mid n[\,\circ\,] \Longrightarrow N''$, *and for all processes* P *there is a system context* $C[-]$ *and systems* M' *and* N' *such that* $M'' \bullet P \gtrsim C[M']$, $N'' \bullet P \gtrsim C[N']$, *and* $M' \mathcal{R} N'$;

- *if* $M \xrightarrow{\texttt{*.exit_}n} M''$ *then there exist a system* N'' *such that* $n[\circ \mid N] \Longrightarrow N''$, *and for all processes* P *there is a system context* $C[-]$ *and systems* M' *and* N' *such that* $M'' \bullet P \gtrsim C[M']$, $N'' \bullet P \gtrsim C[N']$, *and* $M' \mathcal{R} N'$.

Theorem 4.3 *If* $\mathcal{R}$ *is a bisimulation up-to context and up-to* $\gtrsim$, *then* $\mathcal{R} \subseteq \approx$.

5 Adding Communication

The basic idea is to have an *output process* such as $\langle E\rangle.P$, which outputs the message E and then continues as P, and an input process $(x)Q$ which on receiving a message binds it to x in Q which then executes; here occurrences of x in Q are bound. Notice that we have synchronous output; as discussed in [34, 29, 4] this is not unrealistic because communication in MA is always local. The syntax of our extended language is given in Table 8. The structural and reduction rules below define the semantics of communication:

$$E.(F.P) \;\equiv\; (E.F).P \qquad \varepsilon.P \;\rightarrow\; P \qquad (x).P \mid \langle M\rangle.Q \;\rightarrow\; P\{^{M}/_{x}\} \mid Q$$

The LTS is extended by the introduction of two new pre-actions (E) for input, $\langle - \rangle$ for output, a new form of concretions $(\boldsymbol{\nu}\tilde{m})\langle E\rangle Q$, and, among others, the rules:

$$\frac{-}{\langle E\rangle.P \xrightarrow{\langle - \rangle} \langle E\rangle P} \qquad \frac{-}{(x).P \xrightarrow{(\boldsymbol{E})} P\{^{E}/_{x}\}} \qquad \frac{P \xrightarrow{\langle - \rangle} (\boldsymbol{\nu}\tilde{m})\langle E\rangle P' \quad Q \xrightarrow{(\boldsymbol{E})} Q' \quad \mathrm{fn}(Q') \cap \{\tilde{m}\} = \emptyset}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{m})(P' \mid Q')}$$

In Table 9 we give all the defining rules that should be added to those of Table 3 and Table 4 to obtain the LTS for the extended calculus. Note that in the structural rules of Table 3 we are now assuming that parallel composition and restriction distribute over the new form of concretions $(\boldsymbol{\nu}\tilde{m})\langle E\rangle Q$ in the same manner as $(\boldsymbol{\nu}\tilde{m})\langle P\rangle Q$. The slightly unusual pre-action for output allows a uniform treatment of extrusion of names.

The proof of Theorem 2.2 can be easily completed to take into account the extended calculus. A consequence of working with MA in two levels is that communication capabilities cannot be observed at top-level. This in turn implies that our bisimulations can be applied to the extended calculus, and all the results of Section 3 and Section 4 hold without modifications.

Theorem 5.1 *In the Message Passing Calculus, 1) late bisimilarity, early bisimilarity, and barbed congruence coincide; 2) the up-to context and expansion proof technique is sound.*

6 Algebraic Theory

In this section we prove a collection of algebraic properties using our bisimulation proof methods. Then, we prove the correctness of a protocol for controlling access through a *firewall*, first proposed in [7].

We briefly comment on the laws of Theorem 6.1. We recall that M, N range over systems and P, Q, R over processes. The first two laws are two examples of local communication within private ambients without interference. The third law is the well-known perfect firewall law. The following four laws represent non-interference properties about movements of private ambients. Finally, the last two laws say when opening cannot be interfered.

Theorem 6.1

1. $(\boldsymbol{\nu}n)n[\langle W\rangle.P \mid (x).Q \mid M] \;\cong\; (\boldsymbol{\nu}n)n[P \mid Q\{^{W}/_{x}\} \mid M]$ *if* $n \notin \mathrm{fn}(M)$
2. $(\boldsymbol{\nu}n)n[\langle W\rangle.P \mid (x).Q \mid \prod_{j\in J} \texttt{open_}k_j.R_j] \;\cong\; (\boldsymbol{\nu}n)n[P \mid Q\{^{W}/_{x}\} \mid \prod_{j\in J} \texttt{open_}k_j.R_j]$
3. $(\boldsymbol{\nu}n)n[P] \;\cong\; \mathbf{0}$ *if* $n \notin \mathrm{fn}(P)$

4. $(\boldsymbol{\nu} n)((\boldsymbol{\nu} m)m[\texttt{in_}n.P] \mid n[M]) \;\cong\; (\boldsymbol{\nu} n)n[(\boldsymbol{\nu} m)m[P] \mid M]$ *if* $n \notin \mathrm{fn}(M)$

5. $(\boldsymbol{\nu} m, n)(m[\texttt{in_}n.P] \mid n[\prod_{j\in J} \texttt{open_}k_j.R_j]) \;\cong\; (\boldsymbol{\nu} m, n)n[m[P] \mid \prod_{j\in J} \texttt{open_}k_j.R_j]$

6. $(\boldsymbol{\nu} n)n[(\boldsymbol{\nu} m)m[\texttt{out_}n.P] \mid M] \;\cong\; (\boldsymbol{\nu} n)((\boldsymbol{\nu} m)m[P] \mid n[M])$ *if* $n \notin \mathrm{fn}(M)$

7. $(\boldsymbol{\nu} n)n[m[\texttt{out_}n.P] \mid \prod_{j\in J} \texttt{open_}k_j.R_j] \;\cong\; (\boldsymbol{\nu} n)(m[P] \mid n[\prod_{j\in J} \texttt{open_}k_j.R_j])$ *if* $m \neq k_j$, *for* $j \in J$

8. $n[(\boldsymbol{\nu} m)(\texttt{open_}m.P \mid m[N]) \mid Q] \;\cong\; n[(\boldsymbol{\nu} m)(P \mid N) \mid Q]$ *if* $Q \equiv M \mid \prod_{j\in J}\langle W_j\rangle.R_j$ *and* $m \notin \mathrm{fn}(N)$

9. $(\boldsymbol{\nu} n)n[(\boldsymbol{\nu} m)(\texttt{open_}m.P \mid m[Q]) \mid R] \;\cong\; (\boldsymbol{\nu} n)n[(\boldsymbol{\nu} m)(P \mid Q) \mid R]$ *if* $R \equiv \prod_{i\in I}\langle W_i\rangle.S_i \mid \prod_{j\in J} \texttt{open_}k_j.R_j$ *and* $m, n \notin \mathrm{fn}(Q)$.

Proof [Sketch] The proofs of the above laws, except for (3) and (9), are by exhibiting the appropriate bisimulation. In all cases the bisimulation has a similar form, $\mathcal{S} = \{(LHS, RHS)\} \cup \approx$, where LHS, RHS denote the left hand side, right hand side respectively of the identity, parameterised over names, processes and systems. For proving the laws (3) and (9) we need to show that the above $\mathcal{S}$ is a bisimulation up to context and up to structural congruence. We illustrate the proof of the law (3). Let $\mathcal{S} = \{((\boldsymbol{\nu} n)n[Q], \mathbf{0}) \mid \forall Q \text{ s.t. } n \notin \mathrm{fn}(Q)\}$. We show that $\mathcal{S}$ is a bisimulation up to context and up to structural congruence. The most delicate cases are those regarding the silent moves $*.\texttt{enter_}k$ and $*.\texttt{exit_}k$. For instance, if

$$(\boldsymbol{\nu} n)n[P] \xrightarrow{*.\texttt{enter_}k} (\boldsymbol{\nu} n)k[\circ \mid n[P']] \equiv k[\circ \mid (\boldsymbol{\nu} n)n[P']]$$

then

$$\mathbf{0} \mid k[\,\circ\,] \Rrightarrow\equiv k[\circ \mid \mathbf{0}]$$

and up to context and structural congruence we are still in $\mathcal{S}$. □

Crossing a firewall A protocol is discussed in [7] for controlling access through a firewall. The ambient w represents the firewall; the ambient m is a trusted agent containing a process Q that is supposed to cross the firewall. The firewall ambient sends into the agent a pilot ambient k with the capability $\texttt{in_}w$ for entering the firewall. The agent acquires the capability by opening k. The process Q carried by the agent is finally liberated inside the firewall by the opening of ambient m. Names m and k act like passwords which guarantee the access only to authorised agents. Here is the protocol in MA:

$$AG \stackrel{\text{def}}{=} m[\texttt{open_}k.(x).x.Q] \qquad FW \stackrel{\text{def}}{=} (\boldsymbol{\nu} w)w[\texttt{open_}m.P \mid k[\texttt{out_}w.\texttt{in_}m.\langle\texttt{in_}w\rangle]]$$

The correctness (of a slight variant) of the protocol above is shown in [7] for may-testing [9] proving that $(\boldsymbol{\nu} m, k)(AG \mid FG) \;\cong\; (\boldsymbol{\nu} w)w[Q \mid P]$ under the conditions that $w \notin \mathrm{fn}(Q)$, $x \notin \mathrm{fv}(Q)$, $\{m, k\} \cap (\mathrm{fn}(P) \cup \mathrm{fn}(Q)) = \emptyset$. The proof relies on non-trivial contextual reasonings. We show how it can be established using our bisimulation proof methods.

The system on the right can be obtained from that one on the left by executing six τ-actions. So, it suffices to prove that $\cong$ is insensitive to all these τ-actions. The result follows from the algebraic laws of Theorem 6.1 and the following two laws:

Lemma 6.2 *Let P, Q, and R be processes. Then*

1. $(\boldsymbol{\nu} k, m, w)(k[\texttt{in_}m.P] \mid m[\texttt{open_}k.Q] \mid w[\texttt{open_}m.R]) \cong (\boldsymbol{\nu} k, m, w)(m[k[P] \mid \texttt{open_}k.Q] \mid w[\texttt{open_}m.R])$

2. $(\boldsymbol{\nu} m, w)(m[\langle\texttt{in_}w\rangle \mid (x).P] \mid w[\texttt{open_}m.Q]) \cong (\boldsymbol{\nu} m, w)(m[P\{^{\texttt{in_}w}/_x\}] \mid w[\texttt{open_}m.Q])$

Theorem 6.3 *If $w \notin \mathrm{fn}(Q)$, $x \notin \mathrm{fv}(Q)$, and $\{m, k\} \cap (\mathrm{fn}(P) \cup \mathrm{fn}(Q)) = \emptyset$, then $(\boldsymbol{\nu} m, k)(AG \mid FG) \cong (\boldsymbol{\nu} w)w[Q \mid P]$.*

Proof It suffices to apply the algebraic laws of Theorem 6.1 and Lemma 6.2. More precisely, we apply, in sequence, Law (7) of Theorem 6.1, Law (1) of Lemma 6.2, Law (9) of Theorem 6.1, Law (2) of Lemma 6.2, and Laws (5) and (9) of Theorem 6.1. □

7 Related Work

Higher-order LTSs for Mobile Ambients can be found in [6, 13, 33, 10]. But we are not aware of any form of bisimilarity defined using these LTSs. A simple first-order LTS for MA without restriction is proposed by Sangiorgi in [29]. Using this LTS the author defines an *intensional* bisimilarity for MA that separates terms on the basis of their internal structure.

Our work is the natural prosecution of [18] where an LTS and a labelled characterisation of reduction barbed congruence are given for a variant of Levi and Sangiorgi's Safe Ambients, called SAP. The main differences with respect to [18] are the following:

- SAP differs from MA for having co-capabilities and passwords, both features are essential to prove the characterisation result in SAP.
- Our env-actions, unlike those in [18], are truly late, as they do not mention the process provided by the environment. This process can be added *later*, when playing the bisimulation game.
- Our actions for ambient's movement, unlike those in SAP, report the name of the migrating ambient. For instance, in $k.\texttt{enter_}n$ we say that ambient k enters n. 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.
- Co-capabilities also allow the observation of the movement of an ambient whose name is private. As a consequence, the perfect firewall equation does not hold neither in SAP, nor in Safe Ambients. In MA, the movements of an ambient whose name is private cannot be observed. This is why the perfect firewall equation holds.

Apart from [18], other forms of bisimilarity for higher-order calculi, such as Distributed π-calculus [14], Seal [34], Nomadic Pict [32], and a Calculus for Mobile Resources [12], can be found in [20, 8, 32, 12, 5], but only [20, 12, 5] prove labelled characterisations of a contextually defined notion of equivalence. The perfect firewall equation as already been proved for Morris-style contextual equivalence in [13] using a context lemma.

Finally, we believe that interesting labelled characterisations of typed reduction barbed congruence for MA can be derived along the lines of [20], enhancing the algebraic laws of Section 6.

Acknowledgements The authors would like to thank Vladimiro Sassone who spotted a problem in an early draft of the paper. The anonymous referees contributed useful comments. Francesco Zappa Nardelli is grateful to the Foundations of Computing Group of University of Sussex, for the kind hospitality and support. He is partly funded by 'MyThS: Models and Types for Security in Mobile Distributed Systems', EU FET-GC IST-2001-32617.

References

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

[2] S. Arun-Kumar and M. Hennessy. An efficiency preorder for processes. *Acta Informatica*, 29:737–760, 1992.

[3] G. Boudol. Asynchrony and the π-calculus. Technical Report RR-1702, INRIA-Sophia Antipolis, 1992.

[4] M. Bugliesi, G. Castagna, and S. Crafa. Boxed ambients. In *Proc. 4th TACS*, volume 2215 of *LNCS*. Springer-Verlag, 2001.

[5] M. Bugliesi, S. Crafa, M. Merro, and V. Sassone. Communication interference in mobile boxed ambients. Forthcoming Technical Report. An extended abstract appeared in Proc. FSTTCS'02, LNCS, Springer-Verlag.

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

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

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

[9] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[10] G. Ferrari, U. Montanari, and E. Tuosto. A LTS semantics of ambients via graph synchronization with mobility. In *Proc. ICTCS*, volume 2202 of *LNCS*, 2001.

[11] C. Fournet and G. Gonthier. A hierarchy of equivalences for asynchronous calculi. In *Proc. 25th ICALP*, pages 844–855, 1998.

[12] J.C. Godskesen, T. Hildebrandt, and V. Sassone. A calculus of mobile resources. In *Proc. 10th CONCUR '02*, volume 2421 of *LNCS*, 2002.

[13] A. D. Gordon and L. Cardelli. Equational properties of mobile ambients. *Journal of Mathematical Structures in Computer Science*, 12:1–38, 2002.

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

[15] K. Honda and M. Tokoro. An Object Calculus for Asynchronous Communications. In *Proc. ECOOP '91*, volume 512 of *LNCS*. Springer Verlag, 1991.

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

[17] F. Levi and D. Sangiorgi. Controlling interference in ambients. In *Proc. 27th POPL*. ACM Press, 2000.

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

[19] M. Merro and F. Zappa Nardelli. Bisimulation proof methods for mobile ambients. Computer Science Report 2003:01, `http://cogslib.cogs.susx.ac.uk/csr_abs.php?cs`, University of Sussex, 2003.

[20] M. Hennessy M. Merro and J. Rathke. Towards a behavioural theory of access and mobility control in distributed system. To appear in Proc. 5th FoSSaCS '03, LNCS, 2003, Springer-Verlag.

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

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

[23] R. Milner and D. Sangiorgi. Barbed bisimulation. In *Proc. 19th ICALP*, volume 623 of *LNCS*, pages 685–695. Springer Verlag, 1992.

[24] D.M. Park. Concurrency on automata and infinite sequences. In P. Deussen, editor, *Conf. on Theoretical Computer Science*, volume 104 of *LNCS*. Springer Verlag, 1981.

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

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

[27] D. Sangiorgi. Locality and non-interleaving semantics in calculi for mobile processes. *Theoretical Computer Science*, 155:39–83, 1996.

[28] D. Sangiorgi. On the bisimulation proof method. *Journal of Mathematical Structures in Computer Science*, 8:447–479, 1998.

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

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

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

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

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

[34] J. Vitek and G. Castagna. Seal: A framework for secure mobile computations. In *Internet Programming Languages*, 1999.

A Appendix

Table 8 The Message-passing Mobile Ambients in Two Levels

Names: $a, b, \ldots, k, l, m, n, \ldots \in \mathbf{N}$

Capabilities:

C	::=	$\texttt{in_}n$	may enter into n
	$\mid$	$\texttt{out_}n$	may exit out of n
	$\mid$	$\texttt{open_}n$	may open n

Expressions:

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

Guards:

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

Systems:

M, N	::=	$\mathbf{0}$	termination
	$\mid$	$M_1 \mid M_2$	parallel composition
	$\mid$	$(\boldsymbol{\nu} n)M$	restriction
	$\mid$	$n[P]$	ambient

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

Table 9 Labelled Transition System - Communication

Pre-actions: $\pi ::= \ldots \mid (E) \mid \langle - \rangle$

Concretions: $K ::= (\boldsymbol{\nu}\tilde{m})\langle P \rangle Q \mid (\boldsymbol{\nu}\tilde{m})\langle E \rangle Q$

(π Output) $\dfrac{-}{\langle E \rangle.P \xrightarrow{\langle - \rangle} \langle E \rangle P}$ (π Input) $\dfrac{-}{(x).P \xrightarrow{(E)} P\{^{E}/_{x}\}}$ (π Path) $\dfrac{E.(F.P) \xrightarrow{\pi} Q}{(E.F).P \xrightarrow{\pi} Q}$

(τ Eps) $\dfrac{-}{\epsilon.P \xrightarrow{\tau} P}$ (τ Comm) $\dfrac{P \xrightarrow{\langle - \rangle} (\boldsymbol{\nu}\tilde{m})\langle E \rangle P' \quad Q \xrightarrow{(E)} Q' \quad \mathrm{fn}(Q') \cap \{\tilde{m}\} = \emptyset}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{m})(P' \mid Q')}$