ON REDUCTION SEMANTICS FOR THE PUSH AND PULL AMBIENT CALCULUS

Iain Phillips
Maria Grazia Vigliotti
Department of Computing, Imperial College, London
{iccp,mgv98}@doc.ic.ac.uk

Abstract Honda and Yoshida showed how to obtain a meaningful equivalence on processes in the asynchronous π-calculus using equational theories identifying insensitive processes. We apply their approach to a dialect of Cardelli and Gordon's Mobile Ambients. The version we propose here is the Push and Pull Ambient Calculus, where the operational semantics is no longer defined in terms of ambients entering and exiting other ambients, but in terms of being pulled and being pushed away by another ambient. Like the standard ambient calculus, this dialect has the computational power of the asynchronous π-calculus. We contend that the new calculus allows a better modelling of certain security issues.

1. Introduction

The Ambient Calculus (Mobile Ambients, or MA) has been introduced by Cardelli and Gordon as a new model for distributed mobile computation that would take into consideration the reality of the World Wide Web, where division into administrative domains requires handling the notions of both mobile code and authorisation [3]. Ambients, which are meant to represent administrative domains, have a tree structure possibly containing sub-ambients; the notion of access and mobility is captured by the operational semantics where processes equipped with the appropriate capability can freely enter or exit an ambient. The calculus has become in a short time very popular [4, 2, 1, 13] (just to cite a few papers); in particular a great deal of research has been done on types and security. Moreover different dialects have been proposed [10, 11].

On the semantics for the calculus there has been fairly modest progress. Some techniques for proving contextual equivalence have been developed [7], but in terms of bisimulation, the only successful definition has been developed for a dialect of MA, safe ambients with passwords (SAP) [11], and not for the standard calculus (to the best of our knowledge), despite different attempts [6, 15]. The techniques developed in [11] for SAP depend on the particular features of SAP; therefore these are not applicable to standard MA.

Another possible way to define equivalence is to work with *barbed bisimulation*, relying on the definition of reduction relation and an observational predicate, or *barb* [12, 9, 14]. However with barbed bisimulation one needs to use the power of contexts to get a congruence, and one must also commit a priori to a notion of barb, when in certain calculi there might be several possible choices.

Honda and Yoshida [9] have proposed a general formulation of process semantics which induces a canonical congruence based solely on the reduction relation and equational reasoning. Notice that the equivalence obtained is a congruence and we do not mention barbs. In this paper we present the results of applying Honda-Yoshida methods to a variant of MA. The starting point is the definition of *insensitive processes*, which are processes that do not interact with the environment. We then define sound theories. A theory is *sound* if it identifies insensitive and structurally equivalent processes, is consistent, and is preserved by contexts and by the reduction relation. We will see that these minimal requirements are enough to induce a meaningful equivalence, *contextual barbed bisimulation* over the class of processes. In the asynchronous π-calculus setting, contextual barbed bisimulation and barbed congruence coincides [5]. In our setting it remains future work to prove this.

Notice that we do not have to commit in advance to any kind of observation—any observable becomes a property with which the theory is equipped a posteriori. Using the Honda-Yoshida techniques, the observables are indeed the names of the top level ambients, confirming the original choice of barbs made by Cardelli and Gordon.

The Honda-Yoshida approach can be applied to standard MA, but it can be applied to other dialects. The particular dialect we present here is the Push and Pull Ambient Calculus (PAC). The philosophy of PAC is different from MA. The difference lies in the control of the movements of the ambient. In the pure AC, ambients are seen as agents that can enter and exit other administrative domains if equipped with the right capability, but which are not in control of their own boundaries. Other ambients might cross their boundaries. In the following example we show that an ambient called Client, willing to enter into a server (Server), can do that if it is in possession of the name of the Server.

$$\mathbf{Client}\boxed{EnterServer} \mid \mathbf{Server}\boxed{Program}$$

reduces to

$$\mathbf{Server}\boxed{\mathbf{Client}\boxed{EnteredServer} \mid Program}$$

Notice that in this setting, Client decides to enter Server, and Server cannot send Client away in the case that Client is a malicious agent. Similarly for exit. In the PAC setting, administrative domains have total control of their own boundaries, in the sense that an ambient can pull in other ambients and push away unwanted or dangerous agents.

The scenario above would be rewritten as follows:

$$\mathbf{Client}\boxed{Program} \mid \mathbf{Server}\boxed{PullClient \mid Program}$$

reduces to

$$\mathbf{Server}\boxed{\mathbf{Client}\boxed{Program} \mid PulledClient \mid Program}$$

The reduction relation is very similar to standard MA, but in this case the *control* of movements of the ambient lies in the server and not in the client. This new dialect is as powerful as the asynchronous π-calculus and therefore it is Turing complete. There does not seem to exist any obvious encoding from PAC into MA or vice versa.

The main advantage of this new dialect is that we can model interesting scenarios in distributed systems. For instance, consider the scenario of a bank's server with its own LAN, which is an administrative domain. A customer, which is a registered user, can gain access to some services offered by the bank. Even though the customer might not gain access to private or classified administrative sub-domains, it can nevertheless misbehave, for instance by consuming too much time or resources of the service and depriving other customers of the use of the same service. In the standard ambient world, unless the customer leaves, there is no way in which the main server can send away the problematic customer. However this is very easy to represent using the new primitives of PAC. The server deals with a request call from the customer, it allows the customer to gain access to the services, but if the customer misbehaves, the server can always usher the customer away and call another customer.

The main contribution of the paper is a novel approach to the semantics of ambients, in the unlabelled setting, using the techniques developed in [9]. We will obtain contextual bisimulation starting from a minimal set of equations involving insensitive processes. We also propose a different version of MA, where the capabilities are changed in such a way that the control of the movements of ambients lies within the boundaries of the administrative domain. We believe this version helps towards the modelling of problems of network security.

The rest of the paper is organised as follows: in Section 2 we will introduce the new primitives for the Push and Pull Ambient Calculus and its operational semantics based on the reduction relation. In Section 3 we will introduce the notion of insensitive processes that will play a crucial role for the main development of the paper. In Section 4 we will present sound theories and how to define a meaningful congruence based solely on equations; the main result is Theorem 1. In Section 5 we give the operational characterisation of the congruence as contextual bisimulation (Theorem 2) and in Section 6 we will see that our theory is independent from the definition of insensitive processes. In Section 7 we discuss an encoding of the asynchronous π-calculus into our calculus, the main result being Theorem 4. Finally we give conclusions and discuss future work.

2. Syntax and reductions of the Push and Pull Ambient Calculus (PAC)

In this section we are going to introduce the primitives for the Push and Pull Ambient Calculus. In comparison with the Mobile Ambients (MA) of Cardelli and Gordon [3], two different capabilities are introduced: $\mathbf{push}_m\, n$ and $\mathbf{pull}_m\, n$ instead of $\mathbf{in}\, n$ and $\mathbf{out}\, n$; the rest of the syntax remains unchanged. Capability $\mathbf{push}_m\, n$ is the same as the objective move $mv\ n\ out\ m$ briefly mentioned in [3], but $\mathbf{pull}_m\, n$ has no such counterpart. We assume there is an infinite set of names $\mathcal{N}$, ranged over by $n, m, \ldots$.

The set of processes $\mathcal{P}$ is given by the following grammar:

$$m\,[\mathbf{pull}_m\, n.P \mid Q] \mid n\,[R] \to m\,[P \mid Q \mid n\,[R]]$$
$$m\,[\mathbf{push}_m\, n.P \mid n\,[Q] \mid R] \to m\,[P \mid R] \mid n\,[Q]$$
$$\mathbf{open}\, n.Q \mid n\,[R] \to Q \mid R$$
$$\langle n\rangle \mid (m)P \to P\{n/m\}$$

$$\frac{P \to Q}{P \mid R \to Q \mid R} \qquad \frac{P \to Q}{(new\ \ n)P \to (new\ \ n)Q}$$

$$\frac{P \to Q}{n\,[P] \to n\,[Q]} \qquad \frac{P \equiv P' \qquad P' \to Q' \qquad Q' \equiv Q}{P \to Q}$$

Figure 1. Reduction Relation

$$P, Q ::= \mathbf{0} \mid !P \mid P|Q \mid (new\ \ n)P \mid n\,[P] \mid M.P \mid (n).P \mid \langle n\rangle$$

where M stands for the capabilities, which are the following:

$$M \quad ::= \quad \mathbf{push}_m\, n \mid \mathbf{pull}_m\, n \mid \mathbf{open}\, n$$

Intuitively **0** stands for the inactive process, $!P$ simulates recursion by spinning off copies of P, $P \mid Q$ is the parallel composition of two processes, $(new\ \ n)P$ (restriction) creates a new name n in P, $n\,[P]$ is the ambient n containing the active process P and $M.P$ is the process P guarded by the capability M. The meaning of the capabilities is intuitively the following: $\mathbf{open}\, n$ dissolves an ambient with name n; $\mathbf{pull}_m\, n$ and $\mathbf{push}_m\, n$, are primitives which can be exercised within an ambient named m only; $\mathbf{pull}_m\, n$ causes an ambient with name n to be pulled inside the current ambient named m; $\mathbf{push}_m\, n$ pushes an ambient with name n out of the current ambient named m. $(n)P$ will input any message in the top-level ambient, and $\langle n\rangle$ is an output with no continuation. This form of communication is more limited than in MA, where sequences of capabilities can be passed; we have adopted this formulation for simplicity, and because this is all we need for the encoding of the asynchronous π-calculus (Section 7). The set of free names of P is written $\mathsf{fn}(P)$ and is defined in the standard way, taking into account that the binding operators are restriction and input.

The *reduction relation* $P \to P'$, is defined in Figure 1. We use $\twoheadrightarrow$ for the reflexive and transitive closure of the reduction relation. Structural congruence $\equiv$ identifies processes that we do not want to tell apart for any semantic reason. It is defined in Figure 2. As ever, if $P \equiv Q$ then $\mathsf{fn}(P) = \mathsf{fn}(Q)$.

Contextual equivalence (also called may testing) equates two processes that admit the same observation in any context. In the following definition we are following Cardelli and Gordon. We can observe the name of a top level ambient, which does not fall in the scope of the restriction as the following definition expresses.

$$
\begin{array}{rcl}
P \mid \mathbf{0} & \equiv & P \\
P \mid Q & \equiv & Q \mid P \\
(P \mid Q) \mid R & \equiv & P \mid (Q \mid R) \\
(new\ \ n)\mathbf{0} & \equiv & \mathbf{0} \\
(new\ \ m)(new\ \ n)P & \equiv & (new\ \ n)(new\ \ m)P \\
(new\ \ n)(P \mid Q) & \equiv & P \mid (new\ \ n)Q \ \text{ if } n \notin \mathsf{fn}(P) \\
(new\ \ m)n\,[P] & \equiv & n\,[(new\ \ m)P] \ \text{ if } n \neq m \\
!P & \equiv & P \mid\ !P \\
!\mathbf{0} & \equiv & \mathbf{0}
\end{array}
$$

Figure 2. Structural Congruence

Definition 1 (Strong and weak barbs) *A process P* exhibits ambient n*, written as $P \downarrow n$, iff $P \equiv (new\ \ p_1 \ldots p_n)(n\,[P'] \mid P'')$ and $n \notin \{p_1 \ldots p_n\}$. The predicate $P \Downarrow n$ means $P \twoheadrightarrow \downarrow n$.*

A *context* is a process $\mathcal{C}\{\}$ with zero or more occurrences of a hole. $\mathcal{C}\{P\}$ denotes the result of filling each occurrence of a hole in $\mathcal{C}$ with a copy of P, allowing the possible capture of free names. We shall also need contexts with *different* holes, producing $\mathcal{C}\{P_1, \ldots, P_k\}$ in an obvious fashion.

Definition 2 *Processes P, Q are* contextually equivalent *($P \simeq Q$) if for all names n and all contexts $\mathcal{C}$, $\mathcal{C}\{P\} \Downarrow n$ if and only if $\mathcal{C}\{Q\} \Downarrow n$.*

2.1. Examples

There is no obvious way to encode Cardelli and Gordon's MA in PAC (or vice versa). In this section we give some examples to show that in our calculus some basic computation is possible, such as choice. In Section 7 we shall see that we can encode the asynchronous π-calculus.

Cardelli and Gordon discuss a scenario of 'mobile agent authentication' [3]. They point out that a process at the top level inside an ambient is 'privileged', in that it can directly affect the movement of the ambient, and can open subambients. Care must be taken that unauthorised processes do not achieve a privileged position. Although PAC handles movement somewhat differently, the idea of top-level processes being privileged is equally true.

Cardelli and Gordon show how to model a top-level process which wishes to leave the surrounding ambient (named $home$) and then return to resume a top-level position. We can do this quite straightforwardly in PAC as follows:

$$home\,[(new\ \ n)(\mathbf{push}_{home}\, n.\mathbf{pull}_{home}\, n.\mathbf{open}\, n \mid n\,[P])]$$

A *firewall* in PAC is simply a restricted ambient $(new\ \ n)n\,[P]$. Thus a firewall prepared to admit trusted ambients named m is $(new\ \ n)n\,[!\mathbf{pull}_n\, m]$. This is rather simpler than in MA, precisely because we have chosen to let control reside with the admitting ambient rather than the entering ambient.

Finally we show how to encode choice. This was done by Cardelli and Gordon and ours is a modification of what is present in [3].

$$\begin{array}{rcl} n \Rightarrow P + m \Rightarrow Q & \stackrel{\mathrm{df}}{=} & (new\ \ pqr) \\ & & (p\ [\mathbf{pull}_p\, n.\mathbf{push}_p\, n.(\mathbf{push}_p\, q \mid q\ [\mathbf{open}\, r.P])] \mid \\ & & (p\ [\mathbf{pull}_p\, m.\mathbf{push}_p\, m.(\mathbf{push}_p\, q \mid q\ [\mathbf{open}\, r.Q])]) \mid \\ & & \mathbf{open}\, q \mid r\ [\ ]) \end{array}$$

In the presence of $n\ [R]$ and assuming that p, q, r are not free in P, Q, R, we have that:

$$n \Rightarrow P + m \Rightarrow Q \mid n\ [R] \twoheadrightarrow\simeq P \mid n\ [R]$$

3. Insensitive processes

An insensitive process is one which can never react with its environment. In the setting of the asynchronous π-calculus Honda and Yoshida [9] define the active names $\mathsf{an}(P)$ of a process P:

$$\begin{array}{rcl} \mathsf{an}(a(b).P) & = & \{a\} \\ \mathsf{an}(\overline{a}\langle b\rangle) & = & \{a\} \end{array}$$

(together with the clauses for replication, parallel, restriction). A name a is *active* in P if P is immediately able to engage in communication on channel a. A process P is *insensitive* if $\mathsf{an}(P') = \emptyset$ for all P' such that $P \twoheadrightarrow P'$.

It is rather harder to define active names for the Ambient Calculus. Clearly n is active in $n\ [P]$ as well as $\mathbf{open}\, n.P$. But we also have to allow that n is active in $m\ [\mathbf{pull}_m\, n.P]$.

Definition 3 *The set of* active names *is defined as follows:*

$$\begin{array}{rclrcl} \mathsf{an}(\mathbf{0}) & = & \emptyset & \mathsf{an}((new\ \ n)P) & = & \mathsf{an}(P) - \{n\} \\ \mathsf{an}(\mathbf{pull}_m\, n.P) & = & \{m, n\} & \mathsf{an}(P \mid Q) & = & \mathsf{an}(P) \cup \mathsf{an}(Q) \\ \mathsf{an}(\mathbf{push}_m\, n.P) & = & \{m, n\} & \mathsf{an}(!P) & = & \mathsf{an}(P) \\ \mathsf{an}(\mathbf{open}\, n.P) & = & \{n\} & \mathsf{an}((n)P) & = & \{\star\} \\ \mathsf{an}(n\ [P]) & = & \{n\} \cup (\mathsf{an}(P) - \{\star\}) & \mathsf{an}(\langle n\rangle) & = & \{\star\} \end{array}$$

Here $\star$ is a special 'name' representing possible communication at the top level.

It is straightforward that $\mathsf{an}(P) \subseteq \mathsf{fn}(P) \cup \{\star\}$ for any P; also that if $P \equiv Q$ then $\mathsf{an}(P) = \mathsf{an}(Q)$, and that if $P \downarrow n$ then $n \in \mathsf{an}(P)$. We are now ready to give the definition of insensitive processes.

Definition 4 (cf [9]) *A process P is* insensitive *if $\mathsf{an}(P') = \emptyset$ for all P' such that $P \twoheadrightarrow P'$. We let U, V range over insensitive processes.*

Example 1 *Examples of insensitive processes:*

- $(new\ \ n)(\mathbf{open}\, n.P)$;

- $(new\ \ m, n)(\mathbf{pull}_m\, n.P)$;
- $(new\ \ k, p_1 \dots p_n)k\,[P]$ *where* $\mathsf{fn}(P) \subseteq \{p_1 \dots p_n\}$

A *renaming* is a function $\sigma : \mathcal{N} \to \mathcal{N}$. We denote by $P\sigma$ the result of renaming the free names of P according to σ. We conventionally extend renamings to active names by letting $\sigma(\star) = \star$. Renamings can introduce extra reactions if they identify two distinct names. However, we can show that insensitive processes remain insensitive after renaming, which will be useful later on.

Proposition 1 *If U is insensitive and σ is a renaming, then $U\sigma$ is insensitive.* $\square$

The following lemma tells us that insensitive processes never interact with the surrounding environment.

Lemma 1 *Let P be a process such that $\mathsf{an}(P) = \emptyset$ and let $\mathcal{C}\{\}$ be a context. If $\mathcal{C}\{P\} \to Q$ then either $Q \equiv \mathcal{C}\{P'\}$ where $P \to P'$, or $Q \equiv \mathcal{C}'\{P, P\sigma\}$ (some $\mathcal{C}', \sigma$) where $\mathcal{C}\{R\} \to \mathcal{C}'\{R, R\sigma\}$ (any R).* $\square$

The statement of Lemma 1 has to involve contexts with renamings and two different holes because of examples such as $\langle m \rangle \mid !(n)P \to P\{m/n\} \mid !(n)P$.

4. Sound theories

Honda and Yoshida showed how to obtain an equivalence on processes starting from a very basic set of equations [9]. We are going to apply the same method to the PAC. The main idea is to define sound theories in general, and then consider the union of all such theories.

We shall consider equational theories, which are simply sets of pairs $\langle P, Q \rangle$ of processes, from which we can deduce equations between processes by the usual laws of equational reasoning. As usual, a theory is *consistent* if not all processes are identified in the theory.

We shall need to impose some conditions of *soundness* on theories in order that they capture behavioural equivalence. These conditions include the identification of insensitive processes and *reduction* closure, meaning that equality is preserved through reduction. We will see these minimal requirements are powerful enough to define a meaningful congruence over processes.

First we fix the notation. We let $\mathcal{E}$ range over theories. We will say that $\mathcal{E} \vdash P = Q$ if $P = Q$ is derivable in $\mathcal{E}$, and $\mathcal{E} \not\vdash P = Q$ otherwise. We will write $P = Q, P \neq Q$ when it is clear from the context which particular theory we are referring to.

Definition 5 *A theory is* reduction closed *if whenever $P = Q$ and $P \twoheadrightarrow P'$ then there exists a Q' such that $Q \twoheadrightarrow Q'$ and $P' = Q'$.*

Definition 6 *A theory is* sound *if:*

- *it contains structural congruence;*
- *it is consistent;*

- *it is reduction closed;*
- *it identifies all the insensitive processes.*

We shall deem two processes to be equivalent if they are equated in some sound theory. In order for this to make sense, we need to show that sound theories exist, and that the union of all sound theories is itself sound. We first establish that sound theories exist.

Definition 7 *Let $\mathcal{E}_{Ins}$ be the theory generated by $\equiv$ and identifying all the insensitive processes.*

We shall see that $\mathcal{E}_{Ins}$ is sound.

Definition 8 *An equational theory $\mathcal{E}$ preserves* strong (weak) barbs *if whenever $\mathcal{E} \vdash P = Q$, then for every name n, if $P \downarrow n$ then $Q \downarrow n$ (if $P \downarrow n$ then $Q \Downarrow n$).*

Lemma 2 *$\mathcal{E}_{Ins}$ preserves strong barbs.* $\square$

Clearly a theory $\mathcal{E}$ that preserves strong (weak) barbs must be consistent since $\mathcal{E} \not\vdash n\,[\,] = \mathbf{0}$ (any n). Finally we are ready to prove the result:

Proposition 2 *$\mathcal{E}_{Ins}$ is sound.*

Proof By construction, $\equiv$ is included and insensitive processes are identified. Consistency follows from Lemma 2. So it remains to show reduction closure.

In the case of $\equiv$ it follows from the definition of $\rightarrow$. In the case of the laws identifying the insensitive processes, suppose that $\mathcal{C}\{U\} = \mathcal{C}\{V\}$. Then by Lemma 1 we have that if $\mathcal{C}\{U\}$ reacts (in one step) it is either $C\{\ \}$ that reacts to produce $\mathcal{C}\{U, U\sigma\}$, which can be imitated by $\mathcal{C}\{V, V\sigma\}$ (note that $U\sigma$ and $V\sigma$ are insensitive by Proposition 1); or it is U which reacts, $U \rightarrow U'$, and U' is still insensitive, in which case $\mathcal{C}\{V\} \twoheadrightarrow \mathcal{C}\{V\}$. $\square$

So far we have proved that there is at least one sound theory, namely $\mathcal{E}_{Ins}$, which is the *minimal* sound theory according to our definition. A sound theory in general may be more generous, in the sense that it may equate more processes; therefore it is not trivial or automatic that *any* sound theory $\mathcal{E}$ preserves weak barbs. Indeed this is the core of the paper, whose proof relies heavily on the fact that a sound theory is consistent, unlike the proof of Proposition 2, where consistency was a consequence of the preservation of the barbs. We need the following lemma, which holds for any sound theory.

Lemma 3 (Incompatible pairs [9]) *Let $\mathcal{E}$ be a sound theory. Then, $n\,[\,] \neq \mathbf{0}$, all n.*

Proof Let's assume by contradiction that for some m we have that $m\,[\,] = \mathbf{0}$. First of all we are going to prove that for all n, $n\,[\,] = \mathbf{0}$.

Now $(new\ \ m)(\mathbf{open}\, m.n\,[\,] \mid m\,[\,]) = (new\ \ m)(\mathbf{open}\, m.n\,[\,] \mid \mathbf{0})$, then:

$$(new\ \ m)(\mathbf{open}\, m.n\,[\,] \mid m\,[\,]) \rightarrow n\,[\,]$$

and $(new\ \ m)(\mathbf{open}\, m.n\,[\,]\ \mid\ \mathbf{0})$ is insensitive. By reduction closure we have that $n\,[\,] = \mathbf{0}$.

We now prove that $\mathcal{E}$ is inconsistent and thereby obtain a contradiction. Take any process P, and take m such that $m \notin \mathsf{fn}(P)$. We know from above that $m\,[\,] = \mathbf{0}$. So

$$(new\ \ m)(\mathbf{open}\, m.P \mid m\,[\,]) = (new\ \ m)(\mathbf{open}\, m.P \mid \mathbf{0})\ .$$

Moreover, because $(new\ \ m)(\mathbf{open}\, m.P \mid \mathbf{0})$ is insensitive we have

$$(new\ \ m)(\mathbf{open}\, m.P \mid m\,[\,]) = \mathbf{0}\ .$$

By reduction closure we have that $P = \mathbf{0}$, so that the theory is inconsistent. Contradiction! So $m\,[\,] \neq \mathbf{0}$ after all, for any m. $\Box$

The following result is very important, because it tells us that any sound theory is a posteriori equipped with observables.

Proposition 3 *Let $\mathcal{E}$ be a sound theory. Then it preserves weak barbs.*

Proof Assume $\mathcal{E} \vdash P = Q$ and $P \downarrow n$. Let $A = \mathsf{fn}(P) \cup \mathsf{fn}(Q)$. Take $m, k \notin A$. Now $(new\ \ A, k)k\,[P \mid \mathbf{open}\, n.(\mathbf{push}_k\, m \mid m\,[\,])] \twoheadrightarrow (new\ \ A, k)(k\,[P']) \mid m\,[\,]$. By reduction closure $(new\ \ A, k)k\,[Q \mid \mathbf{open}\, n.(\mathbf{push}_k\, m \mid m\,[\,])] \twoheadrightarrow Q'$ with

$$(new\ \ k)(k\,[P']) \mid m\,[\,] = Q'\ .$$

Since $(new\ \ A, k)(k\,[P'])$ is insensitive, $m\,[\,] = Q'$. If $\mathbf{open}\, n.(\mathbf{push}_k\, m \mid m\,[\,])$ is no longer a subprocess of Q' then immediately $Q \Downarrow n$. So suppose $\mathbf{open}\, n.(\mathbf{push}_k\, m \mid m\,[\,])$ is still inside Q'. Then $Q' \equiv (new\ \ A, k)k\,[Q'' \mid \mathbf{open}\, n.(\mathbf{push}_k\, m \mid m\,[\,])]$. But it is not hard to see that Q' is insensitive, giving $m\,[\,] = \mathbf{0}$. Contradiction of Lemma 3. So $Q \Downarrow n$. $\Box$

Theorem 1 *Let $\mathcal{E}_m$ be the union of all sound theories. Then $\mathcal{E}_m$ is sound.*

Proof It is easy to see that the union of any set of sound theories is reduction closed and contains the set of insensitive processes and structural congruence. For consistency we use Proposition 3. It is easy to see that the union of any set of sound theories also preserves weak barbs, and is therefore consistent. $\Box$

We will see in the next section that there is an operational characterisation of the maximal sound theory, namely contextual barbed bisimulation.

5. Contextual barbed bisimulation

In this section we are going to give a more operational definition of the union of all the sound theories. This goes under the name of *contextual barbed bisimulation*. This equivalence relation, that is a congruence, holds only between equal processes, giving an operational meaning to the syntactic relation $\mathcal{E} \vdash P = Q$ defined in the previous section.

Definition 9 *A symmetric relation $\mathcal{S}$ is a* contextual barbed bisimulation *if whenever $P \; \mathcal{S} \; Q$ then for any n and contexts $\mathcal{C}$:*

- *if $\mathcal{C}\{P\} \downarrow n$ then $\mathcal{C}\{Q\} \Downarrow n$;*
- *if $\mathcal{C}\{P\} \rightarrow P'$ then for some Q' we have $\mathcal{C}\{Q\} \twoheadrightarrow Q'$ and $P' \; \mathcal{S} \; Q'$*

Two processes are said to be contextual barbed equivalent *($P \approx Q$) if there exists a contextual barbed bisimulation such that $P \; \mathcal{S} \; Q$.*

By standard arguments, $\approx$ is the largest bisimulation, and $\equiv \subseteq \approx \subseteq \simeq$. We can show that $\approx$ coincides with the union of the sound theories as the following theorem states:

Theorem 2 (cf [9]) $\approx = \mathcal{E}_m$. □

6. Adequate sets of insensitive processes

We earlier defined insensitive processes via active names. It is natural to wonder how far our results on sound theories depend on the particular definition of active names. In this section we show that a quite minimal subset of insensitive processes suffices, implying that the results are quite robust. This set is defined independently of active names.

Given a set of insensitive processes T, a theory is *T-sound* if it

- contains structural congruence;
- is consistent;
- is reduction closed;
- identifies all the processes in T.

Clearly a sound theory is T-sound (for any T). The union of all T-sound theories, denoted $\mathcal{E}_m^T$, plainly includes $\mathcal{E}_m$. We shall say that a set T is 'adequate' if $\mathcal{E}_m^T = \mathcal{E}_m$.

Definition 10 *T_{Open} consists of the following processes:*

- $(new \;\; n)n\;[P]$ *where* $\mathsf{fn}(P) = \emptyset$
- $(new \;\; n)(\mathbf{open}\, n.P)$
- $(new \;\; k)k\;[(new \;\; n)(\mathbf{open}\, n.P \mid Q)]$ *where* $\mathsf{fn}(Q) \subseteq \{n\}$ *and* $Q \not\Downarrow n$

The heart of the matter when choosing sets of insensitive processes is whether we can prove the next proposition, that allows us to build the maximal sound theory.

Proposition 4 *Let $\mathcal{E}$ be a T_{Open}-sound theory. Then $\mathcal{E}$ preserves weak barbs.*

Proof Follow the proof of Proposition 3, noting that only the axioms in T_{Open} are used. □

Theorem 3 *T_{Open} is adequate: $\mathcal{E}_m^{T_{Open}} = \mathcal{E}_m$.*

Proof We know $\mathcal{E}_m \subseteq \mathcal{E}_m^{T_{Open}}$. So it is enough to show that $\mathcal{E}_m^{T_{Open}}$ is sound. $\mathcal{E}_m^{T_{Open}}$ is consistent by Proposition 4. It is reduction closed, being the union of reduction closed theories. It contains all insensitive processes since $\mathcal{E}_m \subseteq \mathcal{E}_m^{T_{Open}}$. □

7. Encoding the asynchronous π-calculus

In this section we show that the asynchronous π-calculus can be encoded in our version of the ambient calculus, which guarantees that the PAC has not only the computational power of the asynchronous π-calculus, but is also Turing complete.

In [3] Cardelli and Gordon gave an encoding of the asynchronous π-calculus [8] into their MA with communication. The idea is that for each communication channel m, an ambient named m is created. Processes wishing to either input or output on channel m enter ambient m, where they communicate. Then the input process leaves m in order to continue its execution. In order to effect this, both input and output processes need to be encased in an ambient name io which is known to the channel ambient, so that they can be opened up and enabled to communicate. Thus the encoding requires a new public name.

We present an encoding into the PAC which follows quite different lines. The idea is that an output on channel m is an ambient named m which contains a value. These output ambients exist in the 'ether' represented by the top level in the tree of ambients. Communication occurs when an output ambient is pulled in by a restricted ambient representing an input on m. It then dissolves its own enclosing ambient before continuing its execution. It will be the case that barbs on outputs (which are what are normally considered in the asynchronous π-calculus) correspond exactly to barbs for ambients. The encoding seems simpler than the one in [3].

The π-calculus is often augmented with the *matching* operator $[m = n]P$. Structural congruence is then augmented with the rule $[m = n]P \equiv P$. We can define a *derived* matching operator in PAC. The basic idea is that **open** $m \mid n\,[\,]$ can react iff $m = n$. We insulate this potential reaction from the outside world inside a restricted ambient, so that no other processes can react with either of the two sides.

$$[m = n]P \stackrel{\text{df}}{=} (new\ \ cd)(c\,[\mathbf{open}\,m.\mathbf{push}_c\,d \mid n\,[\,] \mid d\,[\,]\,] \mid \mathbf{open}\,d.\mathbf{open}\,c.P)$$

Matching can also be simulated in standard MA.

The syntax of the asynchronous π-calculus with matching ($A\pi_=$) is as follows:

$$P, Q ::= \mathbf{0} \mid !P \mid P|Q \mid (new\ \ n)P \mid m(n).P \mid \bar{m}\langle n\rangle \mid [m = n]P$$

We omit the standard definitions of structural congruence and reduction. Barbs in the asynchronous π-calculus are only on outputs (not inputs). We shall need contextual barbed equivalence ($\approx$) and contextual equivalence ($\simeq$) on $A\pi_=$. We omit the definitions, which are much as for PAC.

Our encoding of $A\pi_=$ into PAC is as follows:

$$\begin{array}{rcl}
[\![m(n).P]\!] & = & (new\ \ l, r) \\
 & & (\mathbf{open}\,l \mid r\,[(n).l\,[\mathbf{open}\,r.[\![P]\!]] \mid \mathbf{pull}_r\,m.\mathbf{open}\,m.\mathbf{push}_r\,l]) \\
 & & \text{where } l, r \notin \mathsf{fn}([\![P]\!]) = \mathsf{fn}(P) \\
[\![\bar{m}\langle n\rangle]\!] & = & m\,[\langle n\rangle] \\
[\![\mathbf{0}]\!] & = & \mathbf{0} \\
[\![(new\ \ n)P]\!] & = & (new\ \ n)[\![P]\!] \\
[\![P \mid Q]\!] & = & [\![P]\!] \mid [\![Q]\!] \\
[\![!P]\!] & = & ![\![P]\!] \\
[\![[m = n]P]\!] & = & [m = n][\![P]\!]
\end{array}$$

Note that each π reduction is encoded by six steps. The first of these steps pulls the output inside the input, and commits to the π reduction. The remaining steps then proceed in a deterministic fashion, and cannot be interfered with, since they take place inside restriction.

Proposition 5 *If $P \twoheadrightarrow Q$ holds in $A\pi_=$, then $[\![P]\!] \twoheadrightarrow\approx [\![Q]\!]$.* $\square$

Lemma 4 *Let $P \in A\pi_=$. If $[\![P]\!] \twoheadrightarrow R$ then there is an $A\pi_=$ process Q such that $P \twoheadrightarrow Q$ and $[\![Q]\!] \approx R$. Moreover, if $R \downarrow n$ then $Q \downarrow n$.* $\square$

Lemma 4 does not hold for Cardelli and Gordon's encoding, since the io ambients are not restricted, and are consumed in stages during execution, so that there are intermediate states which do not correspond under any reasonable notion of equivalence to the encoding of a π process.

Lemma 5 *Let $P \in A\pi_=$ and let n be a name. Then $P \Downarrow n$ iff $[\![P]\!] \Downarrow n$.* $\square$

Theorem 4 *Let P and Q be $A\pi_=$ processes.*

1. *If $[\![P]\!] \approx [\![Q]\!]$ then $P \approx Q$.*
2. *If $[\![P]\!] \simeq [\![Q]\!]$ then $P \simeq Q$.*

Proof By Lemmas 4 and 5. $\square$

The converse results do not hold, since in $A\pi_=$ we have $m(n).\bar{m}\langle n\rangle \approx \mathbf{0}$, whereas $[\![m(n).\bar{m}\langle n\rangle]\!]$ is not equivalent to $[\![\mathbf{0}]\!]$ even under $\simeq$.

We do not know whether Theorem 4 holds for Cardelli and Gordon's translation. However our method of proof would not be available in their setting, since we make use of Lemma 4.

8. Conclusions

In this paper we have presented the Push and Pull version of Mobile Ambients, that allows us to model some interesting problems in distributed settings, which seem hard (or impossible) to model in standard MA. We have presented a novel approach to the semantics, which consists of applying semantic techniques developed in [9]. The results presented here could be easily recast for standard MA, but it remains to see whether this approach would equally work for other dialects such as SAP [11], where new syntactical constructs are present.

Acknowledgments

We would like to thank Martin Berger, Cedric Fournet, Andy Gordon, Philippa Gardner, Kohei Honda, Sergio Maffeis and Andrew Phillips for useful discussions. We also thank the anonymous referees for helpful comments. Maria Grazia Vigliotti is supported by an *EPSRC* grant.

References

[1] T. Amtoft, A.J. Kfoury, and S.M. Pericas-Geertsen. What are polymorphically-typed ambients? In *Proceedings of ESOP'01*, volume 2028 of *Lectures Notes in Computer Science*, pages 206–220. Springer-Verlag, 2001.

[2] L. Cardelli, G. Ghelli, and A.D. Gordon. Mobility types for Mobile Ambients. In *Proceedings of ICALP'99*, volume 1644 of *Lectures Notes in Computer Science*, pages 230–239. Springer-Verlag, 1999.

[3] L. Cardelli and A.D. Gordon. Mobile ambients. In *Proceedings of FoSSaCS'98*, volume 1378 of *Lectures Notes in Computer Science*, pages 140–155. Springer-Verlag, 1998. Also Theoretical Computer Science, 240(1), 177–213, 2000.

[4] L. Cardelli and A.D. Gordon. Types for Mobile Ambients. In *Proceedings of POPL '99*, pages 79–92. ACM, January 1999.

[5] C. Fournet and G. Gonthier. A hierarchy of equivalences for asynchronous calculi. In *Proceedings of ICALP'98*, volume 1443 of *Lectures Notes in Computer Science*, pages 844–855. Springer-Verlag, 1998.

[6] A.D. Gordon and L. Cardelli. Technical annex. Available at: www.adg.com/, 1998.

[7] A.D. Gordon and L. Cardelli. Equational Properties of mobile ambients. In *Proceedings of FoSSaCS'99*, volume 1578 of *Lectures Notes in Computer Science*, pages 212–226. Springer-Verlag, 1999.

[8] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In *Proceedings of ECOOP '91*, volume 512 of *Lectures Notes in Computer Science*, pages 133–147. Springer-Verlag, 1991.

[9] K. Honda and N. Yoshida. On the reduction-based process semantics. *Theoretical Computer Science*, 151:437–486, 1995.

[10] F. Levi and D. Sangiorgi. Controlling Interference for Ambients. In *Proceedings of POPL'00*, pages 352–364. ACM, 2000. Extended version available at: www-sop.inria.fr/meije/personnel/Davide.Sangiorgi.

[11] M. Merro and M. Hennessy. Bisimulation congruences in Safe Ambients. In *Proceedings of POPL'02*. ACM, 2002. To appear.

[12] R. Milner and D. Sangiorgi. Barbed Bisimulation. In *Proceedings of ICALP*, volume 623 of *Lectures Notes in Computer Science*, pages 685–695. Springer-Verlag, 1992.

[13] H.R. Nielson and F. Nielson. Validating firewalls in mobile ambients. In *Proceedings of CONCUR'99*, volume 1664 of *Lectures Notes in Computer Science*, pages 463–477. Springer-Verlag, 1999.

[14] D. Sangiorgi. *Expressing mobility in Process Algebra: First-Order and Higher-Order Paradigms*. PhD thesis, University of Edinburgh, 1993.

[15] M.G. Vigliotti. Transition Systems for the Ambient Calculus. Master's thesis, Imperial College, 1999.