Acta Informatica (2007) 43:535–594
DOI 10.1007/s00236-007-0038-z

ORIGINAL ARTICLE

Fair ambients

Yuxi Fu

Received: 9 May 2006 / Accepted: 10 January 2007 /
Published online: 28 February 2007

Abstract Based on an analysis of the capability operators of the Calculus of Mobile Ambients, three fairness principles are proposed to safeguard the interactions of the ambients. The Calculus of Fair Ambient is designed to meet these fairness principles. A labeled transition semantics for the calculus is defined to support structural investigation. The bisimulation theory of the fair ambients is studied and two coincidence results are established. An expressiveness result of the calculus is formally established by proving that it contains the pi calculus as a sub-calculus.

1 Capability, anonymity and mobility

The Calculus of Mobile Ambients, MA for short, was introduced and studied in [8–11] by Cardelli and Gordon as a language that formalizes the mobility of computing [6]. The central notion of the calculus is that of ambient, a piece of program residing in a specified location. The introduction of the ambients captures several features in distributed/mobile computing. The first is that computations are scattered over spatially distributed locations. The second is that computations happen in a parallel fashion in the sense that each program at a location can be executed on its own. The third is that computations are mobile, meaning that a piece of program might migrate from one place to another and that programs can be executed in transit. In MA the syntax for an ambient is as follows:

$$a[P]$$

where a is the ambient name indicating the location, and P is the ambient body specifying the computation. To define mobility the Calculus of Mobile Ambients must be equipped with operators that define how a piece of program migrates from a source

Y. Fu (✉)
BASICS, Department of Computer Science,
Shanghai Jiaotong University, Shanghai 200030, China
e-mail: fu-yx@cs.sjtu.edu.cn

ambient to a target ambient. So computations look like this:

$$a[P] \mid b[Q] \xrightarrow{\tau} a[P'] \mid b[M \mid Q']$$

Here the idea is that the program M has moved from the ambient a to the ambient b. It is clear that the above interaction is between two ambients. The authors of MA have made an important design decision that the above transportation is broken down to two more primitive actions:

$$a[P] \mid b[Q] \xrightarrow{\tau} a[P'] \mid M' \mid b[Q] \xrightarrow{\tau} a[P'] \mid b[M \mid Q']$$

Here the ambient $a[P]$ emits to the environment the process M', and then M' enters the ambient $b[Q]$. In these two step actions M', the program in transit, can be observed by other ambients in the environment. If one thinks of the ambients as first class citizens, it is natural that an observable migrating program, like the M' above, should be an ambient. This requirement is also conducive to the idea that a moving object must be somehow protected. The updated view on the decomposition is then as follows:

$$a[P] \mid b[Q] \xrightarrow{\tau} a[P'] \mid c[M'] \mid b[Q] \xrightarrow{\tau} a[P'] \mid b[c[M] \mid Q']$$

In terms of the *ambient* operators, this means that there should be at least two combinators, one induces an exit operation and the other an entrance operation. For that purpose Cardelli and Gordon introduced two action primitives *in* and *out*, the capabilities. In what follows we take a detailed look at the two primitives.

1.1 Capability

Using the *in* and *out* primitives, Cardelli and Gordon defined the semantics of MA by the following reductions:

$$b[in\, a.Q \mid N] \mid a[P] \xrightarrow{\tau} a[b[Q \mid N] \mid P] \tag{1}$$

$$a[b[out\, a.Q \mid N] \mid P] \xrightarrow{\tau} b[Q \mid N] \mid a[P] \tag{2}$$

There are two observations concerning (1) and (2). One is that the entrance operations are symmetric to the exit operations. Notice that both an entrance and an exit have effect on the ambient and the environment. In (1) the ambient $b[in\, a.Q \mid N]$ is part of the environment that is consumed by the ambient $a[P]$, whereas in (2) the ambient $a[b[out\, a.Q \mid N] \mid P]$ releases $b[Q \mid N]$ onto the environment. The transportation of the ambient in (1) and that in (2) goes in the opposite directions. The second observation is that the movement of an ambient is activated by itself. This is called *Subjective Movement* in [11]. In (1) the ambient $b[in\, a.Q \mid N]$ injects itself into $a[P]$. The action is caused from within $b[in\, a.Q \mid N]$. The target ambient can only accept. In (2) the ambient $b[out\, a.Q \mid N]$ escapes from an ambient at a. The action is triggered by the former, the latter can only let go, and the environment can only accept.

There is an obvious reciprocal to Subjective Movement: The movement of an ambient is activated by the environment (*Objective Movement*). By Objective Movement, one could define the operational semantics as follows:

$$b[Q] \mid a[\overline{in}\, b.P \mid L] \xrightarrow{\tau} a[b[Q] \mid P \mid L] \tag{3}$$

$$a[b[Q] \mid \overline{out}\, b.P \mid L] \xrightarrow{\tau} b[Q] \mid a[P \mid L] \tag{4}$$

In (3) the ambient $a[\overline{in}\,b.P \mid L]$ imports an ambient named b, whereas in (4) the ambient $a[b[Q] \mid \overline{out}\,b.P \mid L]$ exports an ambient named b. In both occasions the ambient $b[Q]$ acts passively. The movement is triggered from outside $b[Q]$. The Ambient Calculus with the operational semantics defined by (3) and (4) is essentially the PAC studied by Phillips and Vigliotti in [31]. In PAC the operators $\overline{in}$ and $\overline{out}$ are denoted respectively by *pull* and *push*.

Subjective Movement, and its duality, is subject to debates. In literature some discussions on MA have focused on the lack of controlling power over interactions. In (1) the entrance is made without any permission from the target ambient $a[P]$. In (2) the release is granted with no regards to the environment. Variants of MA have been proposed to increase the controls over interactions. Let's take a look at some measures that have been put forth. All of them exercise some degree of access control.

– One way is to limit the number of entrances to an ambient as well as the number of release from an ambient. This is achieved by adding the capabilities $\overline{in}$ and $\overline{out}$ and define the operational semantics as follows:

$$b[in\,a.Q \mid N] \mid a[\overline{in}.P \mid L] \xrightarrow{\tau} a[b[Q \mid N] \mid P \mid L] \tag{5}$$

and

$$a[b[out\,a.Q \mid N] \mid \overline{out}.P \mid L] \xrightarrow{\tau} b[Q \mid N] \mid a[P \mid L] \tag{6}$$

In (5) the entrance of $b[Q \mid N]$ into a is permitted since $a[\overline{in}.P \mid L]$ has more room to accept ambients. Similarly the prefix $\overline{out}$ says that $a[b[out\,a.Q \mid N] \mid \overline{out}.P \mid L]$ has the potential to release ambients. So an ambient $a[P]$ may specify the number of entrance and exit it allows. This variant is basically Levi and Sangiorgi's Calculus of Safe Ambients (SA) studied in [20]. In Levi and Sangiorgi's notations, the new primitives take the forms $\overline{in}\,a.P$ and $\overline{out}\,a.P$ respectively. But the following operational semantics points out that the difference is largely syntactical.

$$b[in\,a.Q \mid N] \mid a[\overline{in}\,a.P \mid L] \xrightarrow{\tau} a[b[Q \mid N] \mid P \mid L] \tag{7}$$

and

$$a[b[out\,a.Q \mid N] \mid \overline{out}\,a.P \mid L] \xrightarrow{\tau} b[Q \mid N] \mid a[P \mid L] \tag{8}$$

In the notations of Levi and Sangiorgi, the programs $b[\overline{out}\,a.P \mid c[out\,b.Q]]$ and $b[\overline{in}\,a.P]$, for $a \neq b$, could be regarded as containing design errors.

The control on the number of times an ambient's boundary may be crossed can also be exerted dually:

$$b[in.Q \mid N] \mid a[\overline{in}\,b.P \mid L] \xrightarrow{\tau} a[b[Q \mid N] \mid P \mid L] \tag{9}$$

and

$$a[b[out.Q \mid N] \mid \overline{out}\,b.P \mid L] \xrightarrow{\tau} b[Q \mid N] \mid a[P \mid L] \tag{10}$$

The computation defined in (9) makes good sense from a resource sensitive viewpoint. Consider the ambient $!a[in.L]$. If we think of $a[L]$ as a resource that most ambients depend on, then $!a[in.L]$ provides a way for ambients with *different names* to import the resource $a[L]$. We will denote by SA^{d} the Ambient Calculus with the operational semantics defined by (9) and (10).

We mention that MA can be seen as a sub-calculus of SA. A structural translation from MA to SA interprets $a[P]$ as $a[P \mid !\overline{in} \mid !\overline{out}]$. Similarly PAC can be seen as a sub-calculus of SA$^{\mathrm{d}}$ by translating $a[P]$ to $a[P \mid !in \mid !out]$.

– Another way to control the access to ambients is to combine the mechanisms of Subjective Movement and Objective Movement. Now the two parties involved in an interaction must both warrant the interaction as in the following semantics:

$$b[in\,a.Q \mid N] \mid a[\overline{in}\,b.P \mid L] \xrightarrow{\tau} a[b[Q \mid N] \mid P \mid L] \tag{11}$$

and

$$a[b[out\,a.Q \mid N] \mid \overline{out}\,b.P \mid L] \xrightarrow{\tau} b[Q \mid N] \mid a[P \mid L] \tag{12}$$

In (11) the ambient $b[in\,a.Q]$ is capable of entering the ambient a, the latter is willing to give its permission. So the interaction happens. In (12) the ambient $b[out\,a.Q]$ wants to get out of a, and a lets go. The release then happens. This is the ROAM, the Calculus of Robust Ambients, of Guan [18].

– From the point of view of symmetry of interaction the access controls defined in (11) and (12) do not match. In (11) the environment $b[in\,a.N \mid Q]$ enters a after it obtains a permission from a. But in (12) the ambient $b[out\,a.N \mid Q]$ tries to get out of a must get a permission from within a. The environment still has no say over the event. In [22] Merro and Hennessy propose the following operational rule for the semantics of the out capability:

$$a[c[out\,a.Q \mid N] \mid L] \mid \overline{out}\,a.P \xrightarrow{\tau} c[Q \mid N] \mid a[L] \mid P \tag{13}$$

The process $\overline{out}\,a.P$ in (13), the environment, has issued a visa to an ambient from a. In our view (13) is more compatible with (11) than (12). In both rules the accepting sides, the ambient in (11) and the environment in (13), have the final say.

Merro and Hennessy's SAP, the Calculus of Safe Ambients with Passwords, studied in the aforementioned paper adopts the semantics of (5) and (13) with additional gadgets called the passwords.

There is still some incompatibility between (11) and (13). The rule (11) suggests that an environment is a bunch of ambients. The rule (13) says otherwise. This raises the question concerning the citizenship of ambients:

Question 1: Do all interactions happen between ambients?

If we really believe that ambients are first class citizens, it is reasonable for us to assume that all interactions are between ambients. This brings us to the following semantic rule:

$$a[c[out\,a.Q \mid O] \mid L] \mid b[\overline{out}\,a.P \mid N] \xrightarrow{\tau} a[L] \mid c[Q \mid O] \mid b[P \mid N] \tag{14}$$

Comparing (14) to (11), one immediately sees that the pair $in, \overline{in}$ is less error prone than the pair $out, \overline{out}$. For example the two sub-ambients named a in

$$a[in\,b.L_1] \mid a[in\,c.L_2] \mid b[\overline{in}\,a.L_3]$$

does not cause ambiguity. On the other hand, there is no control over the next interaction in the following system

$$a[c[out\,a.N_1] \mid d[out\,a.N_2]] \mid b[\overline{out}\,a.N_3]$$

To remedy this we propose that an ambient leaving a host ambient must specify the reason to do that. In other words, it must specify the name of the ambient that calls upon it. This can be formulated by the following rule

$$a[c[out\, b.Q \mid O] \mid L] \mid b[\overline{out}\, a.P \mid N] \xrightarrow{\tau} a[L] \mid c[Q \mid O] \mid b[P \mid N] \tag{15}$$

Now the ambient $c[out\, b.Q \mid O]$ specifies for whom it may move out of a.

Let's now go back to the original motivation, that is to describe a situation in which an ambient moves from one ambient to another. Using the semantics prescribed in (11) and (15) we have the following reductions:

$$\begin{aligned} a[c[out\, b.in\, b.M] \mid P] \mid b[\overline{out}\, a.\overline{in}\, c.Q] &\xrightarrow{\tau} a[P] \mid c[in\, b.M] \mid b[\overline{in}\, c.Q] \\ &\xrightarrow{\tau} a[P] \mid b[c[M] \mid Q] \end{aligned}$$

For ambient B to import ambient A' from ambient A, the ambient B needs to know both the name of A and the name of A'. It is our intuition that (11) and (15) support a more streamlined semantics, both operationally and observationally. They also provide a reasonable answers to questions about anonymity.

1.2 Anonymity

For mobile ambients anonymity is an important issue. Consider the ambient

$$(n)n[P]$$

This is an anonymous ambient since its name is a secret to the outside world. For anonymity there is a basic question on *Anonymous-Entrance/Anonymous-Exit*:

Question 2: To what extent may an anonymous ambient cross the border of another ambient?

Anonymous Entrance is the entrance of an anonymous ambient into another ambient. Anonymous Exit refers to the action in the opposite direction. It is clear from the following reductions that MA and SA admit both Anonymous-Entrance and Anonymous-Exit:

$$\begin{aligned} (b)b[in\, a.Q] \mid a[P] &\xrightarrow{\tau} a[(b)b[Q] \mid P] \\ a[(b)b[out\, a.Q] \mid P] &\xrightarrow{\tau} (b)b[Q] \mid a[P] \\ (b)b[in\, a.Q] \mid a[\overline{in}.P] &\xrightarrow{\tau} a[(b)b[Q] \mid P] \\ a[(b)b[out\, a.Q] \mid \overline{out}.P] &\xrightarrow{\tau} (b)b[Q] \mid a[P] \end{aligned}$$

In MA an ambient may sneak in or out of an ambient without any consent of the latter. In SA an ambient would notice that someone has got in or out but wouldn't know who. PAC and SA^{d} support Anonymous-Exit, as can be seen from the following reductions:

$$\begin{aligned} a[(b)(b[Q] \mid \overline{out}\, b) \mid P] &\xrightarrow{\tau} (b)b[Q] \mid a[P] \\ a[(b)(b[out.Q] \mid \overline{out}\, b) \mid P] &\xrightarrow{\tau} (b)b[Q] \mid a[P] \end{aligned}$$

Similarly ROAM admits Anonymous-Exit. SAP admits Anonymous-Entrance and Anonymous-Exit.

The anonymity is relevant to the extent that the following question should be addressed: If the entrance/exit of an anonymous ambient can not be detected or stopped, how can the entrance/exit of a vicious ambient be detected or stopped?

From the viewpoint of the algebraic theory, an anonymous ambient may or may not be observable. It depends not only on whether the calculus admits Anonymous Entarance/Anonymous Exit, but also on whether an anonymous ambient can be broken from outside and/or from within. This brings us to another issue.

One can think of $(a)a[P]$ as setting up a firewall around P. This view adds another dimension to anonymity. We say that a firewall is protective if it does not let any bad things in; it is preventive if it does not let any good things out. A perfect firewall is both protective and preventive. A further question that can be asked about the anonymity is concerned with *Breaking-In-Firewall/Breaking-Out-Of-Firewall*:

Question 3: To what degree may an ambient cross the border of an anonymous ambient?

If the firewall of $(a)a[P]$ can be broken in, then $(a)a[P]$ looks like a 'black hole' that sucks ambients. If the firewall of $(a)a[P]$ can be broken out, then $(a)a[P]$ appears as a 'white hole' that produces ambients.

MA, SA and ROAM admit Breaking-Out-Of-Firewall. PAC and SA$^{\mathrm{d}}$ support Breaking-In-Firewall and Breaking-Out-Of-Firewall.

If all firewalls are perfect then Anonymous Entarnce/Anonymous Exit does no harm. On the other hand a system of the form

$$(a_1)\cdots(a_n)(a_1[P_1]\,|\,\cdots\,|\,a_n[P_n])$$

with perfect firewalls is rather uninteresting, and very unreal in practice, if it does not produce any ambients that interacts with environments. An interesting scenario arises when a computational system takes the following standard form:

$$(a_1)\cdots(a_n)(b_1[P_1]\,|\,\cdots\,|\,b_m[P_m])$$

Here some $b_i[P_i]$'s are locally protected, some are not. The protected ambients can still interact with the outside world through the unprotected ambients. The exchanged messages have to be evaluated by a third party, which acts as a gateway between the protected ambients and the outside world.

In this paper we focus on the ambients with perfect firewalls for systems of ambients. In other words the following proposition holds, where $\approx$ is an observational equivalence.

Proposition 1.1 (System firewall) $(a)a[n_1[N_1]\,|\cdots|\,n_m[N_m]] \approx \mathbf{0}$.

Notice that in most variants of MA the above proposition fails. In MA for instance one has, assuming $a \notin n(N)$, that

$$(a)a[n[out\, a.N]] \xrightarrow{\tau} (a)a[\mathbf{0}]\,|\,n[N]$$

Notice also that Proposition 1.1, as well as other propositions on ambient firewalls, is concerned with the observational theory of the ambients in a calculus. It is not about the expressiveness of the calculus.

1.3 Three principles

We study in this paper the *Calculus of Fair Ambients* (FA for short). FA satisfies three principles. The first is about the ambient citizenship:

> *First Class Citizenship*: All actions are interactions. All interactions are between ambients.

This principle rules out the reduction rules (2), (4), (6), (8), (10) and (12), since they do not define interactions between ambients. It also disowns the rule (13) as it introduces interactions between an ambient and a non-ambient. The second principle is about the fair play between a client and a server:

> *Authorization*: The two ambients involved in an interaction must obtain the authorizations from each other.

This means that, apart from actions, there should be co-actions. One of the two interacting ambients authorizes a request from the other, the latter authorizes the service by the former. According to the above principle the reduction rules (1) and (3) are banned for no authorizations are given to the entrance. The third principle is about authentication:

> *Authentication*: The two ambients involved in an interaction must know each other's identities.

It means that two interacting ambients know each other's names. If one thinks of it, this is also about fair play. The rules (5), (7) and (9) do not meet this principle. This is because the ambient $a[\overline{in}\,a.P \mid L]$, for example, is not aware of the identity of the ambient that enters it. For similar reason the rule (14) falls short of the requirement.

These principles can be regarded as a set of answers to Questions 1-3 put forth in the previous subsections. They leave with us only one choice for the operational semantics of FA: the one defined by (11) and (15). Notice that FA disowns Breaking-In-Firewall, Breaking-Out-Of-Firewall and Anonymous-Entrance. In the absence of these, the presence of Anonymous-Exit is harmless.

1.4 Mobility

Why do ambients migrate? They migrate because they are the computational resources. In distributed computing, resources are bought and sold. The *in* and *out* operators make for the bargaining of resources.

Now let's take a look at the open operator from these perspectives. In MA the semantics for the open capability is defined as follows:

$$open\,a.Q \mid a[P] \xrightarrow{\tau} Q \mid P \tag{16}$$

The reduction (16) could be criticized in several accounts. To start with the ambient $a[P]$ should have the right to authorize the action. In SA this is taken into consideration. Levi and Sangiorgi have proposed in [20] a secure version of (16):

$$open\,a.Q \mid a[\overline{open}\,a.P \mid O] \xrightarrow{\tau} Q \mid P \mid O \tag{17}$$

Like in (13), the interaction in (17) violates the First-Class-Citizenship Principle. If all interactions are between ambients, then (17) should really be replaced by the following:

$$b[open\, a.Q \mid N] \mid a[\overline{open}\, a.P \mid L] \xrightarrow{\tau} b[Q \mid N] \mid P \mid L \tag{18}$$

According to the the Authorization Principle, (18) can be improved as follows:

$$b[open\, a.Q \mid N] \mid a[\overline{open}\, b.P \mid L] \xrightarrow{\tau} b[Q \mid N] \mid P \mid L \tag{19}$$

The open capabilities defined in (16) through (19) are quite different from the in and out capabilities. The *in* and *out* operations are about mobility, whereas *open* has little to do with mobility. From the point of view of a buyer (19) is not a very economic way of purchasing resources because everybody can make use of $P \mid L$.

Another reason to reject the present open capabilities is that an ambient might evolve into a non-ambient. A system consisting of ambients could turn into anomaly with ambients and non-ambients. Take for instance

$$\begin{aligned} & b[open\, a \mid \overline{in}\, a.N] \mid a[\overline{open}\, b \mid in\, b.L \mid in\, c.L'] \mid c[\overline{in}\, a.O] \\ & \xrightarrow{\tau} b[\overline{in}\, a.N] \mid in\, b.L \mid in\, c.L' \mid c[\overline{in}\, a.O] \end{aligned}$$

Here the open action is more a destruction, or grave interference as termed in [20], than a competition for resources. In the literature this use of the open capabilities has always been avoided of course. The following scenario, couched in the original MA, seems to be the norm in many applications:

$$u[open\, a.N \mid P] \mid a[in\, u.L \mid Q] \xrightarrow{\tau} u[a[L \mid Q] \mid open\, a.N \mid P] \xrightarrow{\tau} u[L \mid Q \mid N \mid P]$$

The ambient u imports the ambient a, and then immediately opens up a. Several points are worth making about this example. First of all the ambient u opens up a for its own use. It doesn't do that for anybody else. This is supported by the fact that the open operation is applied to the ambient a within u. Second the content of the ambient a has moved into u. So mobility is present if an ambient opens up some resource for its own purpose.

The Calculus of Mobile Ambient is a model for distributed and mobile computing. It is reasonable to expect that all the operators of the ambients define mobility. In this paper we propose an open operator that not only satisfies the three principles but also introduces a kind of mobility and a kind of resource relocation. The semantics of our open operator is prescribed by the following reduction:

$$b[open\, a.Q \mid N] \mid a[\overline{open}\, b.P \mid L] \xrightarrow{\tau} b[P \mid L \mid Q \mid N] \tag{20}$$

In other words, an open operation is an interaction of two ambients who know each other's names and authorize each other's actions. The ambient $b[open\, a.Q \mid N]$ terminates the ambient $a[\overline{open}\, b.P \mid L]$ by exhausting the entire internal resources of a for its own use.

Comparing (20) and (19), one notices that the former is a local version of the latter. The global open can simulate the local one. But the local open enjoys the subject reduction property: If M is a system of ambients and $M \xrightarrow{\tau} N$ then N is a system of ambients.

FA is equipped with the local open operator.

1.5 Contribution

The paper makes the following contributions:

- Based on the analysis of the capability operators, we propose an operational semantics for the ambients that is defined purely in terms of a labeled transition system. By exploiting the labeled transition system, we investigate four bisimulation congruence relations. The bisimilarity intends to make the least reference to contexts. The barbed bisimilarity follows the standard definition. The observable bisimilarity and the weak observable bisimilarity are stronger and respectively weaker than the barbed bisimilarity by definition. We prove that the four bisimilarities coincide. These results improve upon those of Merro and Hennessy in that there is no use of passwords. The coincidence results are supported by the Local Observability property stating that the equivalences are for processes the same way as they are for ambients.
- We propose a new operational semantics for communication that conforms to the three principles. The presence of the communication forces all congruence relations to be closed under substitutions of names. We work out the algebraic theory for FA with communication. In particular we prove that the four bisimilarities, tailored to take into account of the communications, coincide.
- We formally prove that the π-calculus is a sub-calculus of FA with communication. We construct a translation from π to FA that not only preserves and reflects the operational semantics but also preserves and reflects the observational semantics. This result reveals the close relationship between π and MA, two well known frameworks of mobile computing.

The approach advocated in this paper is general enough to help understand the observational theories of the other variants of the Ambient Calculus.

The rest of the paper is organized as follows: Sect. 2 defines the operational semantics of FA. Section 3 gives some examples to illustrate the expressiveness of FA. Section 4 introduces the bisimilarity. Section 5 explains the Bisimulation Lemma crucial to the later proofs. Section 6 reinforces the role of the bisimilarity by showing that it is the same as the observable bisimilarity. Section 7 proves that the bisimilarity coincides with the barbed bisimilarity as well as the weak observable bisimilarity. Section 8 studies the communication mechanism for FA. Section 9 establishes the fact that the π-calculus is a sub-calculus of FA. Section 10 highlights some major points. The appendix reviews some definitions and carries out some proofs.

2 Labeled semantics

The semantics of FA is defined purely in terms of a labeled transition system. No structural congruence is preordained. Let $\mathcal{N}$ be the set of names, ranged over by $a, b, c, \ldots$. A finite sequence $x_1, \ldots, x_n$ of names, as well as the set $\{x_1, \ldots, x_n\}$, will be abbreviated to $\widetilde{x}$. The abstract syntax of *process* is defined by the following abstract grammar:

$$\begin{aligned} P := {} & \mathbf{0} \\ & \kappa.P \\ & P \mid P \\ & a[P] \\ & (a)P \\ & !P \end{aligned}$$

where

$$\begin{aligned} \kappa := {} & in\, a \\ & \overline{in}\, a \\ & out\, a \\ & \overline{out}\, a \\ & open\, a \\ & \overline{open}\, a \end{aligned}$$

Here κ stands for a capability. If κ is $in\, a$ ($\overline{in}\, a$, $out\, a$, $\overline{out}\, a$, $open\, a$, $\overline{open}\, a$), we call a the target name of κ and is denoted by $n(\kappa)$. The capabilities have the following readings:

- "$in\, a$" reads "get into a".
- "$\overline{in}\, a$" reads "get a".
- "$out\, a$" reads "get out to reach a".
- "$\overline{out}\, a$" reads "get someone from a".
- "$open\, a$" reads "open up a".
- "$\overline{open}\, a$" reads "be opened up by a".

Processes of the form $a[P]$ or $(a)a[P]$ are called *ambients*, where a is the name of the ambient. Sometimes we say "ambient a", or even "a", to refer to the ambient $a[P]$ when no confusion arises. A process consisting of a collection of ambients is called a system. Formally systems are defined by the following grammar:

$$\begin{aligned} S := {} & \mathbf{0} \\ & S \mid S \\ & a[P] \\ & (a)S \\ & !S \end{aligned}$$

Throughout the paper we stick to the following notational convention:

- $L, M, N, O, P, Q, L', M', N', O', P', Q', \ldots$ stand for processes;
- $S, T, S', T', \ldots$ stand for systems;
- $A, B, A', B', \ldots$ stand for ambients.

Interactions between ambients are higher order. In order not to introduce higher order action labels in the operational semantics, one uses concretions and contexts. The grammar of the concretions is as follows:

$$D := \langle P \rangle P \mid (x)D$$

To define the structural semantics, it is convenient to extend the scope of the composition and the restriction operators to cover also the concretions. Suppose D is $(\widetilde{m})\langle O\rangle P$ and P is not $\mathbf{0}$. Then the extension is defined as follows:

$$\begin{aligned}
D \mid Q &\overset{\text{def}}{=} (\widetilde{m})\langle O\rangle(P \mid Q) \quad \text{if } \widetilde{m} \cap fn(Q) = \emptyset \\
Q \mid D &\overset{\text{def}}{=} (\widetilde{m})\langle O\rangle(Q \mid P) \quad \text{if } \widetilde{m} \cap fn(Q) = \emptyset \\
(x)(\langle O\rangle\mathbf{0}) &\overset{\text{def}}{=} \langle (x)O\rangle\mathbf{0} \\
(x)D &\overset{\text{def}}{=} (x)(\widetilde{m})\langle O\rangle P \quad \text{if } x \in fn(O) \\
(x)D &\overset{\text{def}}{=} (\widetilde{m})\langle O\rangle(x)P \quad \text{if } x \notin fn(O)
\end{aligned}$$

Contexts are environments that can make finite observations. This simple statement suggests two things: (i) There is no need for the replication operator, or recursion, in the contexts; and (ii) There need be only one hole in a context.

Definition 2.1 Static contexts are defined as follows:

- [_] is a static context;
- if $C[\_]$ is a static context then $C[\_] \mid A$, $A \mid C[\_]$, $(x)C[\_]$ are static contexts.

Contexts are defined as follows:

- a static context is a context;
- if $C[\_]$ is a context then $a[C[\_]]$ is a context.

An ambient context at a is a context of the form $C'[a[C''[\_]]]$ where $C'[\_]$ is a context and $C''[\_]$ is a static context. A standard ambient context at a is an ambient context of the form $a[\_ \mid O]$. An ambient context is an ambient context at some name.

We will let $U, U', \ldots$ to range over the set of processes, concretions and contexts.

In the rules defining the operational semantics λ ranges over the action labels defined below:

$$\begin{aligned}
\lambda := {} & \kappa \\
& a[\kappa], \text{ if } \kappa \neq out\, b \text{ for every } b \\
& [out\, b] \\
& a[[out\, b]] \\
& \tau
\end{aligned}$$

An action could be a process action of the form κ, invoked by the capability κ. It could be the τ action indicating an interaction. It could also be an action performed by ambients. The intuition behind the ambient actions is this:

- "$a[in\, b]$" reads "ambient a moves into ambient b".
- "$a[\overline{in}\, b]$" reads "ambient a imports ambient b".
- "$[out\, b]$" reads "the surrounding ambient moves out to reach ambient b".
- "$a[\overline{out}\, b]$" reads "ambient a extracts an ambient out of ambient b".
- "$a[open\, b]$" reads "ambient a opens up ambient b for its internal resource".
- "$a[\overline{open}\, b]$" reads "ambient a is opened up by ambient b and donates its resource to ambient b".
- "$a[[out\, b]]$" reads "an ambient moves out of ambient a to reach ambient b".

Structural Rule

$$\frac{}{\kappa.P \xrightarrow{\kappa} P} \qquad \frac{P \xrightarrow{\lambda} U}{P \,|\, Q \xrightarrow{\lambda} U \,|\, Q} \qquad \frac{P \xrightarrow{\tau} P'}{a[P] \xrightarrow{\tau} a[P']}$$

$$\frac{P \xrightarrow{\lambda} U \quad n \text{ is not in } \lambda}{(n)P \xrightarrow{\lambda} (n)U} \qquad \frac{P \,|\, !P \xrightarrow{\lambda} U}{!P \xrightarrow{\lambda} U}$$

Ambient Rule

$$\frac{P \xrightarrow{in\, b} P'}{a[P] \xrightarrow{a[in\, b]} \langle a[P']\rangle\mathbf{0}} \qquad \frac{P \xrightarrow{\overline{in}\, b} P'}{a[P] \xrightarrow{a[\overline{in}\, b]} a[\_ \,|\, P']} \qquad \frac{P \xrightarrow{out\, b} P'}{a[P] \xrightarrow{[out\, b]} \langle a[P']\rangle\mathbf{0}} \qquad \frac{P \xrightarrow{\overline{out}\, b} P'}{a[P] \xrightarrow{a[\overline{out}\, b]} a[P']}$$

$$\frac{P \xrightarrow{open\, b} P'}{a[P] \xrightarrow{a[open\, b]} a[\_ \,|\, P']} \qquad \frac{P \xrightarrow{\overline{open}\, b} P'}{a[P] \xrightarrow{a[\overline{open}\, b]} \langle P'\rangle\mathbf{0}} \qquad \frac{P \xrightarrow{[out\, b]} (\widetilde{m})\langle A\rangle P'}{a[P] \xrightarrow{a[[out\, b]]} (\widetilde{m})(A \,|\, a[P'])}$$

Interaction Rule

$$\frac{P \xrightarrow{b[in\, a]} (\widetilde{m})\langle B\rangle P' \quad Q \xrightarrow{a[\overline{in}\, b]} C[\_]}{P \,|\, Q \xrightarrow{\tau} (\widetilde{m})(P' \,|\, C[B])} \qquad \frac{P \xrightarrow{b[[out\, a]]} P' \quad Q \xrightarrow{a[\overline{out}\, b]} Q'}{P \,|\, Q \xrightarrow{\tau} P' \,|\, Q'}$$

$$\frac{P \xrightarrow{a[open\, b]} C[\_] \quad Q \xrightarrow{b[\overline{open}\, a]} (\widetilde{m})\langle Q''\rangle Q'}{P \,|\, Q \xrightarrow{\tau} (\widetilde{m})(C[Q''] \,|\, Q')}$$

Fig. 1 The operational semantics of FA

The labeled transition system is defined in Fig. 1. Notice that all symmetric rules are omitted. Most of the rules are self-explanatory. We only make a couple of comments:

- There are three groups of rules. The first explains how the process actions persist through the structural composition. The second introduces the ambient actions. The third defines the interactions between ambients.
- There are three kinds of transitions. One is of the form $P \xrightarrow{\lambda} P'$ for a process P'; the second is of the form $P \xrightarrow{\lambda} D$ for a concretion D; and the third is of the form $P \xrightarrow{\lambda} C[\_]$ for a context $C[\_]$. Here the context plays a role similar to that of abstraction. In [22] the authors use concretions instead of contexts. We use contexts because we do not impose a structural congruence on the syntax of the processes.
- A transition $P \xrightarrow{\lambda} U$ emphasizes two things: One is that P performs an *action* denoted by λ. The other is that the action achieves the *effect* codified by U. Intuitively the transition $P \xrightarrow{\lambda} (\widetilde{m})\langle M\rangle P'$ indicates the followings: (i) P is able to invoke a higher order interaction; (ii) M, the dynamic, is the part to move; (iii) P', the residual, is the part to stay; (iv) M and P' share the local names $\widetilde{m}$. More specifically $P \xrightarrow{a[in\, b]} (\widetilde{m})\langle A\rangle P'$ means that A is to move into an ambient b; and $P \xrightarrow{[out\, b]} (\widetilde{m})\langle A\rangle P'$ that A is to move out to reach ambient b.
- Our concretions is different from those in [22]. Merro and Hennessy have transitions like $P \xrightarrow{\lambda} (\widetilde{m})\langle P''\rangle_n P'$. It indicates that *after* the interaction, not necessarily *before* the interaction, P'' is in ambient n and P' is outside the ambient n. Here the structural aspect of the resulting process is emphasized. Our concretions stress more on the dynamic aspect.
- We have chosen the action label $[out\, b]$ instead of $a[out\, b]$. This is because an ambient can not see inside another ambient. In $c[L] \,|\, b[a[M]]$ for example $c[L]$

can see b but not a. Using the label $a[out\, b]$ would suggest that a was observable, which was not true.

We will use some common notations and abbreviations. For instance we will write $\Longrightarrow$ for the reflexive and transitive closure of $\xrightarrow{\tau}$, and $\overset{\lambda}{\Longrightarrow}$ for $\Longrightarrow\xrightarrow{\lambda}\Longrightarrow$. We will leave out ambients like $\mathbf{0}$, $a[\mathbf{0}]$ and $a[b[\mathbf{0}] \mid c[\mathbf{0}]]$, etc. to improve readability.

Let's take a look at a couple of examples of derivations using the operational rules: One has

$$\cfrac{\cfrac{}{\overline{out}\, a.B \xrightarrow{\overline{out}\, a} B}}{b[\overline{out}\, a.B] \xrightarrow{b[\overline{out}\, a]} b[B]}$$

and

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{out\, b.A \xrightarrow{out\, b} A}}{c[out\, b.A] \xrightarrow{[out\, b]} \langle c[A]\rangle \mathbf{0}}}{(c)c[out\, b.A] \xrightarrow{[out\, b]} \langle (c)c[A]\rangle \mathbf{0}}}{N \mid (c)c[out\, b.A] \xrightarrow{[out\, b]} \langle (c)c[A]\rangle N}}{(x)(N \mid (c)c[out\, b.A]) \xrightarrow{[out\, b]} (x)\langle (c)c[A]\rangle N}}{a[(x)(N \mid (c)c[out\, b.A])] \xrightarrow{a[[out\, b]]} (x)((c)c[A] \mid a[N])}$$

where we have assumed that $x \in fn(A)$. We can then apply the second interaction rule to obtain the following transition

$$a[(x)(N \mid (c)c[out\, b.A])] \mid b[\overline{out}\, a.B] \xrightarrow{\tau} (x)((c)c[A] \mid a[N]) \mid b[B]$$

This example shows why the label $[out\, b]$ is preferred over the label $c[out\, b]$. It also helps to understand the restriction operation on concretions.

In the rest of the paper we will occasionally say something like "let x be a fresh name" or "for a fresh name x", by which we mean that x does not occur in any syntactical objects under consideration.

Before ending this section we state some simple properties to be used later on.

Lemma 1 *The following properties hold:*

(i) *If $P \xrightarrow{a[\overline{in}\, b]} U$ then U is an ambient context at a.*

(ii) *If $P \xrightarrow{a[open\, b]} U$ then U is an ambient context at a.*

3 Examples

This section serves to demonstrate some of the expressive power of FA by giving several examples. Without communication mechanism FA, like MA, is good to express mobility, but not very convenient to code up functional computation. We will see how communications help to promote FA to accommodate the functional computing style in later sections. We make use of the strong bisimilarity $\sim$, defined in Appendix A, in the following examples.

Example 1 (Renaming) The open capability provides an easy way of renaming an ambient. An ambient whose behavior is the same as $a[P]$ could be renamed to $b[P]$ as the following reduction demonstrates:

$$b[open\, a] \mid a[\overline{open}\, b \mid P] \xrightarrow{\tau} b[P]$$

This property is explored in the following reductions

$$\begin{aligned} b[Q \mid (n)(n[open\, a] \mid a[\overline{open}\, n.P] \mid a[open\, n])] &\xrightarrow{\tau} b[Q \mid (n)(n[P] \mid a[open\, n])] \\ &\Longrightarrow b[Q \mid (n)(n[\overline{open}\, a.P'] \mid a[open\, n])] \\ &\xrightarrow{\tau} b[Q \mid a[P']] \end{aligned}$$

The ambient a is made anonymous and is then restored to the original name later on. In the anonymous state the process P carries out computations with full protection. □

Example 2 (Unknown resource and target) Let S be the system

$$(m)(a[in\, m.Q] \mid m[\overline{in}\, a.\overline{open}\, b] \mid b[open\, m.P])$$

Then one has the following reduction:

$$\begin{aligned} S &\xrightarrow{\tau} (m)(m[a[Q] \mid \overline{open}\, b] \mid b[open\, m.P]) \\ &\xrightarrow{\tau} b[a[Q] \mid P] \end{aligned}$$

The ambient $a[in\, m.Q]$ enters a transfer $m[\overline{in}\, a.\overline{open}\, b]$. It is then shipped to a place it doesn't know. The target does not know the name of $a[in\, m.Q]$ either. Both a and b know that they are doing business with a trusted party m. □

Example 3 (Out) Using the out capability one could define $a[b[out.B] \mid A]$ by the following system $(d)(a[b[out\, d.B] \mid A] \mid d[\overline{out}\, a])$, where d is fresh. The following reduction is admissible

$$a[b[out.B] \mid A] \xrightarrow{\tau} b[B] \mid (d)a[A] \sim b[B] \mid a[A]$$

Notice that the ambient $b[out\, d.B]$ might move to another ambient caused by a co-open action of A. But $b[out\, d.B]$ retains the ability to move out. □

In the following examples, *out* will be used as a primitive. When this derived operator is used, it should be understood that the necessary co-out operators are inserted in the right place.

Example 4 (Universal resource) Let N be $n[A \mid open\, a.P] \mid R$, where A is the ambient $a[out.open\, r.\overline{open}\, n]$ and R is the resource defined by $!r[\overline{open}\, a.M]$. There is an obvious reduction sequence

$$\begin{aligned} N &\xrightarrow{\tau} a[open\, r.\overline{open}\, n] \mid n[open\, a.P] \mid R \\ &\xrightarrow{\tau} a[M \mid \overline{open}\, n] \mid n[open\, a.P] \mid R \\ &\xrightarrow{\tau} n[M \mid P] \mid R \end{aligned}$$

Here R provides the universal resource M that can be accessed by ambients named a. The ambient $n[A \mid open\, a.P]$ sends out an envoy with name a whose mission is to fetch a piece of code from R. In FA mediators, or matchmakers, play a heavy role in getting access to universal resources. □

Example 5 (Non-deterministic choice for ambients) Let $d[\overline{open}\,a] + d[\overline{open}\,b]$ be

$$(c)(c[\overline{open}\,d.\overline{open}\,a] \mid c[\overline{open}\,d.\overline{open}\,b] \mid d[open\,c])$$

Then clearly

$$d[\overline{open}\,a] + d[\overline{open}\,b] \xrightarrow{\tau}\sim d[\overline{open}\,a]$$
$$d[\overline{open}\,a] + d[\overline{open}\,b] \xrightarrow{\tau}\sim d[\overline{open}\,b]$$

The nondeterministic choice $a[P] + b[Q]$ is then defined by

$$(d)((d[\overline{open}\,a] + d[\overline{open}\,b]) \mid a[open\,d.P] \mid b[open\,d.Q])$$

It should be remarked that these definable choices are all internal choices. □

Example 6 (Numerals) We introduce the special names $num, dec, op, zero$ for the numerals and *true* for the booleans. The numerals are defined as follows:

$$\underline{0} \overset{\text{def}}{=} num[!\overline{out}\,zero \mid !out\,op \mid !in\,num]$$
$$\underline{n{+}1} \overset{\text{def}}{=} num[\underline{n} \mid !out\,op \mid !in\,num \mid \overline{open}\,dec]$$

And the operations on the numerals are defined as follows:

$$\underline{+1} \overset{\text{def}}{=} num[\overline{in}\,num.(!out\,op \mid !in\,num \mid \overline{open}\,dec)]$$
$$\underline{-1} \overset{\text{def}}{=} (x)(dec[open\,num.\overline{open}\,x] \mid x[open\,dec] \mid op[\overline{out}\,x])$$
$$\underline{?0} \overset{\text{def}}{=} zero[true[out\,num.\ldots]]$$

The definition of the predecessor makes use of a local name to garbage collect unwanted codes. It is easy to check out the following properties:

$$\underline{n} \mid \underline{+1} \xrightarrow{\tau}\sim \underline{n{+}1}$$
$$\underline{n{+}1} \mid \underline{-1} \overset{\tau}{\Longrightarrow}\sim \underline{n}$$
$$\underline{0} \mid \underline{?0} \xrightarrow{\tau}\sim \underline{0} \mid true[\ldots]$$

A system wants to test or operate on the numerals may send an envoy to do the job. A better encoding of the numerals is possible using communications. □

Example 7 (Facilitated diffusion) Facilitated diffusion is a process of diffusion where molecules diffuse across membranes with the assistance of transport proteins. Actually transport or transmembrane proteins form channels through the cell membranes. The channel can be opened or closed and molecules pass down a concentration gradient. When the concentration of the molecules in the cell is lower than that in the environment, the channel is opened by the cell and molecules may enter into the cell. This could be described by the following interaction:

$$cell[\overline{in}\;mol \mid P] \mid mol[in\;cell \mid out\;env \mid Q] \xrightarrow{\tau} cell[P \mid mol[out\;env \mid Q]]$$

When the concentration of molecules in the environment is lower than that in the cell, the channel is opened by the environment and the molecules exit from the cell. This could be described by the following interaction:

$$cell[P \mid mol[out\;env \mid Q]] \mid env[\overline{out}\;cell \mid R] \xrightarrow{\tau} cell[P] \mid mol[Q] \mid env[R]$$

See for example [7,33] for more on the process approach to system biology. □

Example 8 (Text encryption) Suppose that the sender S wants to send a text T to the receiver R. The text must be encrypted during transition. Let S be

$$a[A \mid d[in\, k.out\, k.T] \mid k[\overline{in}\ d.out\ b.in\ b]]$$

and R be

$$b[B \mid \overline{out}\ a.\overline{in}\ k \mid k[\overline{out}\, k]]$$

Then

$$\begin{aligned}(k)(S \mid R) &\xrightarrow{\tau} (k)(a[A \mid k[d[out\, k.T] \mid out\ b.in\ b]] \mid b[B \mid \overline{out}\ a.\overline{in}\ k \mid k[\overline{out}\, k]])\\ &\xrightarrow{\tau} (k)(a[A] \mid k[d[out\, k.T] \mid in\ b] \mid b[B \mid \overline{in}\ k \mid k[\overline{out}\, k]])\\ &\xrightarrow{\tau} (k)(a[A] \mid b[B \mid k[d[out\, k.T]] \mid k[\overline{out}\, k]])\\ &\xrightarrow{\tau} (k)(a[A] \mid b[B \mid d[T]])\end{aligned}$$

The sender and the receiver share the key k. The sender first encrypts the text with the key k. The encrypted text leaves the sender and is on its way to the receiver. The receiver then accepts the encrypted text. It decrypts the encrypted text using the key k. □

Example 9 (Firewall) Cardelli and Gordan give in [11] an example of using MA to code up a firewall protocol. We are now adapting the example to FA. The firewall Fw and the Agent ag are defined as follows:

$$\begin{aligned}Fw &\overset{\text{def}}{=} (f)f[k[out.\overline{out}\ ag.\overline{in}\ r.\overline{open}\ f] \mid open\ k.M \mid P]\\ Ag &\overset{\text{def}}{=} ag[r[out\ k.in\ k.R] \mid N]\end{aligned}$$

The system $Fw \mid Ag$ have the following operational behavior:

$$\begin{aligned}Ag \mid Fw &\xrightarrow{\tau} ag[r[out\ k.in\ k.R] \mid N] \mid (f)(k[\overline{out}\ ag.\overline{in}\ r.\overline{open}\ f] \mid f[open\ k.M \mid P])\\ &\xrightarrow{\tau} ag[N] \mid r[in\ k.R] \mid (f)(k[\overline{in}\ r.\overline{open}\ f] \mid f[open\ k.M \mid P])\\ &\xrightarrow{\tau} ag[N] \mid (f)(k[\overline{open}\ f \mid r[R]] \mid f[open\ k.M \mid P])\\ &\xrightarrow{\tau} ag[N] \mid (f)f[r[R] \mid M \mid P]\end{aligned}$$

The system Fw protected by a firewall intends to fetch a piece of resource $r[R]$ from the agent ag. Since nobody is supposed to cross the border of the firewall, the system Fw sends out a legate $k[\overline{out}\ ag.\overline{in}\ r.\overline{open}\ f]$. The agent $ag[r[out\ k.in\ k.R] \mid N]$ knows the watchword of the legate, and is happy to pass the resource to the legate. The latter then uploads the resource to the protected system. □

Example 10 (Membrane fusion) Cellular membrane fusion is well known. It is one of the most common ways for molecules to enter or exit cells in processes like fertilization and viral infection. When two cells fuse together, their membranes come together at one location and create a connection between the cells that allows the exchange of materials between them. Eventually the two membranes form one single, continuous membrane surrounding the contents of both cells. We can describe for example the situation when a virus enters a cell by the following reduction:

$$cell[open\ virus \mid P] \mid virus[\overline{open}\ cell \mid Q] \xrightarrow{\tau} cell[P \mid Q]$$

This is an example of using the local open capability. □

4 Bisimulation

Process equivalences are observational in the sense that two processes are equivalent if no context can detect any difference between them. When a process is put in a context, the latter observes the former by interacting with it. To establish the equivalence of two processes one has to examine them in all contexts. The universal quantification implies that the observational equivalences are congruent. It also means that reasoning about the observational equivalences is not easy. Bisimulation equivalences ([26,28]) are proposed to approximate the observational equivalences with the aim to reduce the reference to contexts.

In view of the above remark, it is clear that the challenge of a bisimulation approach is to define an equivalence relation that meets two goals: (i) its dependency on contexts is minimal; (ii) it makes good sense from the observational point of view. What has to be kept in mind is that ambient interactions are of higher order in nature. An ambient action could send an ambient to the environment. It could also receive an ambient from the environment. When an ambient has left a host ambient, it still shares some local names with the ex-host. This means that from an observational point of view it is wrong to consider the ambient and the ex-host in isolation. So bisimulations for ambients can hardly be without any reference to contexts. But what kind of contexts should we consider? When an ambient has emigrated to another ambient it will be placed in an ambient context of the form $a[C[\_]]$. When an ambient is importing an ambient, the immigrant is to be put in an ambient context of the form $a[C[\_]]$. In both cases $C[\_]$ is a static context. Now the ambient context $a[C[\_]]$ is structurally equivalent to $(\widetilde{x})a[\_ \mid O]$ for some O. With this observation in mind, it should not be difficult to see that we only have to consider contexts of the form $a[\_ \mid O]$, the standard ambient contexts.

We are now in a position to define the bisimulations for the ambients.

Definition 4.1 A symmetric relation $\mathcal{R}$ on processes is a bisimulation if the followings hold whenever $P\mathcal{R}Q$:

1. If $P \xrightarrow{\tau} P'$ then $Q \Longrightarrow Q'\mathcal{R}P'$ for some Q'.
2. If $P \xrightarrow{\kappa} P'$ then $Q \overset{\kappa}{\Longrightarrow} Q'\mathcal{R}P'$ for some Q'.
3. If $P \xrightarrow{\lambda} P'$ and $\lambda \in \{a[\overline{out}\, b], a[[out\, b]] \mid a, b \in \mathcal{N}\}$ then $Q \overset{\lambda}{\Longrightarrow} Q'\mathcal{R}P'$ for some Q'.
4. If $P \xrightarrow{[out\, a]} (\widetilde{m})\langle A\rangle P'$ then for every name d and every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ there exist some $\widetilde{n}, B, Q', Q''$ such that $Q \Longrightarrow\xrightarrow{[out\, a]} (\widetilde{n})\langle B\rangle Q'$ and $(\widetilde{n})(B \mid d[Q' \mid O]) \Longrightarrow Q'' \;\mathcal{R}\; (\widetilde{m})(A \mid d[P' \mid O])$.
5. If $P \xrightarrow{a[in\, b]} (\widetilde{m})\langle A\rangle P'$ then for every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ there exist some $\widetilde{n}, B, Q', Q''$ such that $Q \Longrightarrow\xrightarrow{a[in\, b]} (\widetilde{n})\langle B\rangle Q'$ and $(\widetilde{n})(Q' \mid b[B \mid O]) \Longrightarrow Q'' \;\mathcal{R}\; (\widetilde{m})(P' \mid b[A \mid O])$.
6. If $P \xrightarrow{a[\overline{open}\, b]} (\widetilde{m})\langle M\rangle P'$ then for every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ some $\widetilde{n}, N, Q', Q''$ exist such that $Q \Longrightarrow\xrightarrow{a[\overline{open}\, b]} (\widetilde{n})\langle N\rangle Q'$ and $(\widetilde{n})(Q' \mid b[N \mid O]) \Longrightarrow Q'' \;\mathcal{R}\; (\widetilde{m})(P' \mid b[M \mid O])$.
7. If $P \xrightarrow{a[\overline{in}\, b]} C[\_]$ then for every N satisfying $fn(N) \cap bn(C[\_]) = \emptyset$ there exist some $C'[\_], Q'$ such that $Q \Longrightarrow\xrightarrow{a[\overline{in}\, b]} C'[\_]$ and $C'[b[N]] \Longrightarrow Q' \;\mathcal{R}\; C[b[N]]$.

8. If $P \overset{a[open\, b]}{\longrightarrow} C[\_]$ then for every N satisfying $fn(N) \cap bn(C[\_]) = \emptyset$ there exist some $C'[\_]$, Q' such that $Q \Longrightarrow \overset{a[open\, b]}{\longrightarrow} C'[\_]$ and $C'[N] \Longrightarrow Q' \mathcal{R}\, C[N]$.

The bisimilarity $\approx$ is the largest bisimulation.

Here are some explanations about the definition:

1. Clause (1) is standard for bisimulations.
2. Clause (2) discusses the case when P can perform a capability κ. A capability invokes an ambient relocation. In FA all capabilities are exercised within ambients. A reasonable requirement is $b[P' \mid O]\mathcal{R}b[Q' \mid O]$ for every name b and every process O. We shall see that the weaker condition $P'\mathcal{R}Q'$ is sufficient.
3. Clause (3) implies that the effect of the $a[\overline{out}\, b]$ action or the $a[[out\, b]]$ action is local.
4. Clause (4) deals with the case where the ambient A is to emigrate from an ambient d. To account for all possible structures of the contexts, the ambient A must be considered together with $d[P' \mid O]$.
5. Clause (5) describes the situation that an ambient A is capable of moving into an ambient named b. Without losing generality one could take this ambient to be $b[\_ \mid O]$. Then $b[A \mid O]$ and P' should be considered in combination in simulation.
6. Clause (6) is similar to clause (5) except that the process exported is not an ambient but the internal resources of an ambient.
7. Clause (7) takes into account of the fact that if P can perform the action $a[\overline{in}\, b]$ then the ambients it may import must be of the form $b[N]$.
8. Clause (8) is similar to clause (7) except that the process imported is not an ambient but the internal resources of an ambient.

The bisimulations defined by Definition 4.1 is what one might call *early* bisimulations. This is clear from the clauses (4) through (8) of the definition. A late version of the clause (8) for instance is this:

If $P \overset{a[open\, b]}{\longrightarrow} C[\_]$ then there exists some $C'[\_]$ such that $Q \overset{a[open\, b]}{\Longrightarrow} C'[\_]$ and $C'[N]\ \mathcal{R}\ C[N]$ for every N satisfying $fn(N) \cap bn(C[\_]) = \emptyset$.

We need to define the reductions of context to make sense of the notation $Q \overset{a[open\, b]}{\Longrightarrow} C'[\_]$. But it is intuitively clear. Late bisimulations are desirable since they are simpler. We have not been able to show that the late bisimilarity coincides with the early one.

Let's now see some examples:

- $a[\mathbf{0}] \approx c[a[\mathbf{0}] \mid b[\mathbf{0}]]$. Both are equivalent to $\mathbf{0}$.
- $(a)a[out\, b.P] \not\approx \mathbf{0}$ even if $a \notin fn(P)$. When putting in an ambient context like $c[\_ \mid O]$, the ambient $(a)a[out\, b.P]$ could move out of the context if environments want it.
- $(a)a[n[N] \mid in\, b.M] \approx \mathbf{0}$.
- $(a)a[n[N] \mid out\, b.M] \approx (a)a[out\, b.(n[N] \mid M)]$.
- $(c)a[b[out\, c.A]] \approx \mathbf{0}$ for fresh c. This is because all the ambient $b[out\, c.A]$ can do is to move out of a. But this capability can not be invoked since no environment knows c. In general $(c)C[\kappa.P] \approx C[\mathbf{0}]$ if $n(\kappa) = c$ and c does not occur in $C[\mathbf{0}]$. The equivalence fails in the presence of communication.
- $(d)b[out\, a.in\, d] \approx c[out\, a]$. This is because $(d)b[out\, a.in\, d] \overset{[out\, a]}{\longrightarrow} (d)\langle b[in\, d]\rangle\mathbf{0}$ can be simulated by $c[out\, a] \overset{[out\, a]}{\longrightarrow} \langle c[\mathbf{0}]\rangle\mathbf{0}$, and vice versa.

It goes without saying that the observational equivalences must be equivalence relations. So we need to state the following fact before going further.

Proposition 4.2 *The bisimilarity $\approx$ is an equivalence.*

A further justification of Definition 4.1 is the following congruence theorem, whose proof is subsumed by the proof of Lemma 14.

Theorem 4.3 *Suppose $P \approx Q$. Then the following holds:*

(i) $\kappa.P \approx \kappa.Q$;

(ii) $P \mid R \approx Q \mid R$;

(iii) $(x)P \approx (x)Q$;

(iv) $a[P] \approx a[Q]$;

(v) $!P \approx !Q$.

An often used technique to prove equivalence makes use of the bisimulation-up-to relations. Appendix A recalls two such relations relevant in this paper.

5 Bisimulation lemma

In the observational theory of the process calculi, processes are studied in terms of their external interactional behaviors. The internal interactions, the tau actions, can not be observed per se. But their effects can often be observed. Let's take a look at three situations. The first is when an internal action is of the form

$$P_0 \xrightarrow{\tau} P_1$$

such that P_0 does not have any other actions. In this case it is clear that $P_0 \approx P_1$. The second is the following situation

$$P_2 \xleftarrow{\lambda} P_0 \xrightarrow{\tau} P_1$$

where P_0 can perform at least one other action λ. In this case the difference between P_0 and P_1 can normally be observed if there is a difference. The third situation is

$$P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} P_{n-1} \xrightarrow{\tau} P_n \xrightarrow{\tau} P_0$$

where P_i, for $i \in \{1, \ldots, n\}$, may have other actions apart from the tau action. This case is interesting because of the following fact.

Lemma 2 (O-Lemma) *If $P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} P_{n-1} \xrightarrow{\tau} P_n \xrightarrow{\tau} P_0$ then $P_0 \approx P_1 \approx \ldots \approx P_{n-1} \approx P_n$.*

Why is $P_i \approx P_j$? Because $C[P_i] \xrightarrow{\lambda} U$ can be simulated by $C[P_j] \overset{\tau}{\Longrightarrow} C[P_i] \xrightarrow{\lambda} U$ if P_i participates the action. The following is a consequence of the O-Lemma.

Corollary 5.1 *Suppose $P_0 \rhd_0 P_1 \rhd_1 \cdots P_{n-1} \rhd_{n-1} P_n \rhd_n P_0$ where, for each $i \in \{1, \ldots, n\}$, $\rhd_i$ is either $\Longrightarrow$ or $\approx$. Then $P_0 \approx P_1 \approx \ldots P_{n-1} \approx P_n$.*

A special form of Corollary 5.1 is what we call Bisimulation Lemma.

Lemma 3 (Bisimulation lemma) *If* $P \Longrightarrow P' \approx Q$ *and* $Q \Longrightarrow Q' \approx P$ *then* $P \approx Q$.

The above lemma says that $P \approx Q$ if P and Q are related in the following manner

$$P \Longrightarrow\approx Q$$
$$Q \Longrightarrow\approx P$$

We call it Bisimulation lemma because the statement of the lemma reminds one of the bisimulation properties. It is a general result valid for all the equivalences we are aware of. Bisimulation lemma plays a crucial role in the studies of π and χ carried out in [14] and [13,17] respectively. It also plays a crucial role in this paper.

6 Observable bisimulation

A widely studied bisimulation equivalence is the so called barbed bisimilarity [29], which is based on the notion of barbedness. For ambient calculi the question is how to define the barbedness. There are two ways:

– One is the syntactical approach. This is often used when the underlying operational semantics is presented in a reductional style. In [22,24] for instance, it is defined that P is barbed at a if

$$P \equiv (\widetilde{x})(a[\overline{open}\, a.P' \mid R] \mid O)$$

where $a \notin \widetilde{x}$. Other definitions are possible.

– The other is the observational approach. Barbs are defined in terms of the action labels. This is preferred if a labeled transition system is available. In π-calculus, a barb of a process is a channel through which the process may communicate. In ambient calculi a barb is the name of an ambient that *can act*. Take for instance the process $a[\mathbf{0}]$. From an observational point of view, $a[\mathbf{0}]$ is as good as the inactive process $\mathbf{0}$. Therefore $a[\mathbf{0}]$ does not have any barb.

The two approaches are not always equivalent. In some variants of MA, say SA, the two ambients $a[in\, n]$ and $b[in\, n]$ are barbed equivalent according to the first definition. They are not barbed equivalent according to the second definition. The point is that different process calculi and different variants of a process calculus may have different barbedness.

In this paper we study a bisimulation equivalence that is similar to but more robust than the barbed bisimilarity. The equivalence emphasizes the *observable actions* a process is able to perform at a particular state.

Definition 6.1 A process P is observable at a non tau action λ, noted $P\Downarrow\lambda$, if $P \Longrightarrow\xrightarrow{\lambda} U$ for some U. A process P is observable, notation $P\Downarrow$, if $P\Downarrow\lambda$ for some λ. $P \not\Downarrow$ if not $P\Downarrow$.

Definition 6.2 A relation $\mathcal{R}$ is observable if $P\mathcal{R}Q \Rightarrow \forall\lambda.(P\Downarrow\lambda \Leftrightarrow Q\Downarrow\lambda)$.

With these definitions we may define observable bisimulations as in the next definition.

Definition 6.3 A symmetric relation $\mathcal{R}$ on processes is an observable bisimulation if the following properties hold:

(i) It is observable and is closed under context;

(ii) If $Q\mathcal{R}P \xrightarrow{\tau} P'$ then $Q \Longrightarrow Q'\mathcal{R}P'$ for some Q'.

The observable bisimilarity $\approx_{obs}$ is the largest observable bisimulation.

It should be clear that the bisimilarity $\approx$ is an observable bisimulation. The rest of the section is devoted to the proof that $\approx_{obs}$ is a bisimulation. First we prove a technical lemma.

Lemma 4 *Suppose c is fresh. Then $a[P \mid O] \approx_{obs} a[Q \mid O]$ if either $a[P \mid \overline{in}\, c.O] \approx_{obs} a[Q \mid \overline{in}\, c.O]$ or $a[P \mid \overline{out}\, c.O] \approx_{obs} a[Q \mid \overline{out}\, c.O]$ or $a[P \mid open\, c.O] \approx_{obs} a[Q \mid open\, c.O]$.*

Proof It is clear that the following reduction $a[P \mid \overline{in}\, c.O] \mid c[in\, a] \xrightarrow{\tau} a[P \mid O]$ must be matched up by

$$\begin{aligned} a[Q \mid \overline{in}\, c.O] \mid c[in\, a] &\Longrightarrow a[Q_1 \mid \overline{in}\, c.O] \mid c[in\, a] \\ &\xrightarrow{\tau} a[Q_1 \mid O] \\ &\Longrightarrow a[Q' \mid O'] \\ &\approx_{obs} a[P \mid O] \end{aligned}$$

A crucial observation is that the above reduction sequence can be rearranged as follows:

$$\begin{aligned} a[Q \mid \overline{in}\, c.O] \mid c[in\, a] &\xrightarrow{\tau} a[Q \mid O] \\ &\Longrightarrow a[Q' \mid O'] \\ &\approx_{obs} a[P \mid O] \end{aligned}$$

Consequently

$$a[Q \mid O] \Longrightarrow\approx_{obs} a[P \mid O] \tag{21}$$

Symmetrically

$$a[P \mid O] \Longrightarrow\approx_{obs} a[Q \mid O] \tag{22}$$

It follows from (21), (22) and the Bisimulation lemma (Lemma 3) that $a[P \mid O] \approx_{obs} a[Q \mid O]$. □

The observable bisimulations are not very tractable. Results like the one given below are always useful.

Lemma 5 *If $a[P \mid \overline{open}\, b] \approx_{obs} a[Q \mid \overline{open}\, b]$ for a fresh b then $C[P] \approx_{obs} C[Q]$ for every ambient context $C[\_]$.*

Proof For arbitrary x and O let $C'[\_]$ be the context

$$a[\_ \mid \overline{open}\, b] \mid b[open\, a.\overline{open}\, c] \mid c[open\, b.\overline{open}\, x.\overline{out}\; c] \mid x[open\, c.O] \mid c[c[out\; x]]$$

where c is fresh. Then clearly

$$\begin{aligned} C'[P] &\xrightarrow{\tau} b[P \mid \overline{open}\, c] \mid c[open\, b.\overline{open}\, x.\overline{out}\; c] \mid x[open\, c.O] \mid c[c[out\; x]] \\ &\xrightarrow{\tau} c[P \mid \overline{open}\, x.\overline{out}\; c] \mid x[open\, c.O] \mid c[c[out\; x]] \\ &\xrightarrow{\tau} x[P \mid \overline{out}\; c \mid O] \mid c[c[out\; x]] \\ &\xrightarrow{\tau} x[P \mid O] \end{aligned}$$

Since $C'[P] \approx_{obs} C'[Q]$, the above reduction sequence must be simulated by

$$C'[Q] \overset{\tau}{\Longrightarrow} x[Q' \mid O] \Longrightarrow Q''$$

for some Q'', which could be rearranged as

$$C'[Q] \overset{\tau}{\Longrightarrow} x[Q \mid O] \Longrightarrow x[Q' \mid O] \Longrightarrow Q''$$

Therefore $x[Q \mid O] \Longrightarrow Q'' \approx_{obs} x[P \mid O]$. Similarly one has $x[P \mid O] \Longrightarrow P'' \approx_{obs} x[Q \mid O]$. Thus $x[P \mid O] \approx_{obs} x[Q \mid O]$ by Lemma 3. It is then easy to prove that $C[P] \approx_{obs} C[Q]$. □

We now come to the major result of this section. It relates the bisimilarity of two processes to the observable bisimilarity of the processes.

Theorem 6.4 *$P \approx_{obs} Q$ if and only if $P \approx Q$.*

Proof Let $\mathcal{R}$ be the following relation

$$\{(P, Q) \mid c[P \mid \overline{open}\, d] \approx_{obs} c[Q \mid \overline{open}\, d] \text{ for fresh } d\}$$

We shall prove that $\mathcal{R}$ is a bisimulation. Suppose that $Q \mathcal{R} P$ and P does a λ action. It follows from the assumption $c[P \mid \overline{open}\, d] \approx_{obs} c[Q \mid \overline{open}\, d]$ and Lemma 5 that

$$C[P] \approx_{obs} C[Q] \tag{23}$$

for every ambient context $C[\_]$. We will make extensive use of (23) without explicitly referring to it. The proof is a case analysis on λ.

- $\lambda = \tau$. This case is simple.
- $\lambda = in\, b$ and $P \xrightarrow{in\, b} P'$. Let $C[\_]$ be $c[\_ \mid out\, e.\overline{open}\, d] \mid b[\overline{in}\, c] \mid e[\overline{out}\, b]$, where e is fresh. Then

$$\begin{aligned} C[P] &\xrightarrow{\tau} b[c[P' \mid out\, e.\overline{open}\, d]] \mid e[\overline{out}\, b] \\ &\xrightarrow{\tau} c[P' \mid \overline{open}\, d] \end{aligned}$$

  It follows that the above reduction must be matched up by

$$C[Q] \overset{\tau}{\Longrightarrow} c[Q' \mid \overline{open}\, d] \approx_{obs} c[P' \mid \overline{open}\, d]$$

  Hence $Q \overset{in\, b}{\Longrightarrow} Q' \mathcal{R} P'$.
- $\lambda = \overline{in}\, b$ and $P \xrightarrow{\overline{in}\, b} P'$. For a fresh e, let $C[\_]$ be the context

$$c[\_ \mid \overline{open}\, d] \mid b[in\, c.out\, e] \mid e[\overline{out}\, c]$$

  Then

$$\begin{aligned} C[P] &\xrightarrow{\tau} c[P' \mid b[out\, e] \mid \overline{open}\, d] \mid e[\overline{out}\, c] \\ &\xrightarrow{\tau} c[P' \mid \overline{open}\, d] \end{aligned}$$

  The simulation must be $C[Q] \overset{\tau}{\Longrightarrow} c[Q' \mid \overline{open}\, d] \approx_{obs} c[P' \mid \overline{open}\, d]$. Thus $Q \overset{\overline{in}\, b}{\Longrightarrow} Q' \mathcal{R} P'$.

– $\lambda = out\, b$ and $P \xrightarrow{out\, b} P'$. For a fresh e, let $C[\_]$ be the context

$$c[\_ \mid in\, e.\overline{open}\, d] \mid e[\overline{in}\, c] \mid b[\overline{out}\, e]$$

Then

$$\begin{aligned} C[P] &\xrightarrow{\tau} e[c[P \mid \overline{open}\, d]] \mid b[\overline{out}\, e] \\ &\xrightarrow{\tau} c[P' \mid \overline{open}\, d] \end{aligned}$$

The simulation must be $C[Q] \stackrel{\tau}{\Longrightarrow} c[Q' \mid \overline{open}\, d] \approx_{obs} c[P' \mid \overline{open}\, d]$. Thus $Q \stackrel{out\, b}{\Longrightarrow} Q'\mathcal{R}P'$.

– $\lambda = \overline{out}\, b$ and $P \xrightarrow{\overline{out}\, b} P'$. For a fresh e, let $C[\_]$ be

$$c[\_ \mid \overline{open}\, d] \mid b[e[out\, c.e[out\, e]]] \mid e[\overline{out}\, e]$$

Then

$$\begin{aligned} C[P] &\xrightarrow{\tau} c[P' \mid \overline{open}\, d] \mid e[e[out\, e]] \mid e[\overline{out}\, e] \\ &\xrightarrow{\tau} c[P' \mid \overline{open}\, d] \end{aligned}$$

The simulation must be $C[Q] \stackrel{\tau}{\Longrightarrow} c[Q' \mid \overline{open}\, d] \approx_{obs} c[P' \mid \overline{open}\, d]$. Thus $Q \stackrel{\overline{out}\, b}{\Longrightarrow} Q'\mathcal{R}P'$.

– $\lambda = open\, b$ and $P \xrightarrow{open\, b} P'$. For a fresh e, let $C[\_]$ be

$$c[\_ \mid \overline{open}\, d] \mid b[\overline{open}\, c.open\, e] \mid e[\overline{open}\, c]$$

Then

$$\begin{aligned} C[P] &\xrightarrow{\tau} c[P' \mid open\, e \mid \overline{open}\, d] \mid e[\overline{open}\, c] \\ &\xrightarrow{\tau} c[P' \mid \overline{open}\, d] \end{aligned}$$

The simulation must be $C[Q] \stackrel{\tau}{\Longrightarrow} c[Q' \mid \overline{open}\, d] \approx_{obs} c[P' \mid \overline{open}\, d]$. Thus $Q \stackrel{open\, b}{\Longrightarrow} Q'\mathcal{R}P'$.

– $\lambda = \overline{open}\, b$ and $P \xrightarrow{\overline{open}\, b} P'$. For a fresh e, let $C[\_]$ be

$$e[\_] \mid b[open\, e.\overline{open}\, c.\overline{open}\, d] \mid c[open\, b.\overline{out}\, e] \mid e[e[out\, c]]$$

Then

$$\begin{aligned} C[P] &\xrightarrow{\tau} b[P' \mid \overline{open}\, c.\overline{open}\, d] \mid c[open\, b.\overline{out}\, e] \mid e[e[out\, c]] \\ &\xrightarrow{\tau} c[P' \mid \overline{open}\, d \mid \overline{out}\, e] \mid e[e[out\, c]] \\ &\xrightarrow{\tau} c[P' \mid \overline{open}\, d] \end{aligned}$$

The simulation must be $C[Q] \stackrel{\tau}{\Longrightarrow} c[Q' \mid \overline{open}\, d] \approx_{obs} c[P' \mid \overline{open}\, d]$. Thus $Q \stackrel{\overline{open}\, b}{\Longrightarrow} Q'\mathcal{R}P'$.

– $\lambda = a[\overline{out}\, b]$ and $P \xrightarrow{a[\overline{out}\, b]} P'$. For a fresh e, let $C[\_]$ be

$$c[\_ \mid b[d[out\, a.out\, e]] \mid \overline{open}\, d] \mid e[\overline{out}\, c]$$

Then

$$\begin{aligned} C[P] &\xrightarrow{\tau} c[P' \mid d[out\, e] \mid \overline{open}\, d] \mid e[\overline{out}\, c] \\ &\xrightarrow{\tau} c[P' \mid \overline{open}\, d] \end{aligned}$$

The simulation must be $C[Q] \overset{\tau}{\Longrightarrow} c[Q' \mid \overline{open}\, d] \approx_{obs} c[P' \mid \overline{open}\, d]$. Thus $Q \overset{a[\overline{out}\, b]}{\Longrightarrow} Q'\mathcal{R}P'$.

– $\lambda = a[[out\, b]]$ and $P \overset{a[[out\, b]]}{\longrightarrow} P'$. For a fresh e, let $C[\_]$ be

$$c[\_ \mid b[\overline{out}\, a.out\, e] \mid \overline{open}\, d] \mid e[\overline{out}\, c]$$

Then

$$\begin{aligned} C[P] &\xrightarrow{\tau} c[P' \mid b[out\, e] \mid \overline{open}\, d] \mid e[\overline{out}\, c] \\ &\xrightarrow{\tau} c[P' \mid \overline{open}\, d] \end{aligned}$$

The simulation must be $C[Q] \overset{\tau}{\Longrightarrow} c[Q' \mid \overline{open}\, d] \approx_{obs} c[P' \mid \overline{open}\, d]$. So $Q \overset{a[[out\, b]]}{\Longrightarrow} Q'\mathcal{R}P'$.

– $\lambda = [out\, a]$ and $P \overset{[out\, a]}{\longrightarrow} (\widetilde{m})\langle A\rangle P'$. Let e be fresh and O be a process such that $\{\widetilde{m}\} \cap fn(O) = \emptyset$. Define the context $C[\_]$ by

$$c[d[\_ \mid \overline{out}\, a.O] \mid a[\overline{out}\, d.d[out\, d.out\, e]] \mid \overline{open}\, d] \mid e[\overline{out}\, c]$$

It is clear that

$$\begin{aligned} C[P] &\xrightarrow{\tau} c[(\widetilde{m})(A \mid d[P' \mid \overline{out}\, a.O]) \mid a[d[out\, d.out\, e]] \mid \overline{open}\, d] \mid e[\overline{out}\, c] \\ &\xrightarrow{\tau} c[(\widetilde{m})(A \mid d[P' \mid O]) \mid d[out\, e] \mid \overline{open}\, d] \mid e[\overline{out}\, c] \\ &\xrightarrow{\tau} c[(\widetilde{m})(A \mid d[P' \mid O]) \mid \overline{open}\, d] \end{aligned}$$

The simulation of the above reduction sequence can be rearranged as follows:

$$\begin{aligned} C[Q] &\Longrightarrow C[Q'] \\ &\xrightarrow{\tau} c[(\widetilde{n})(B \mid d[Q' \mid \overline{out}\, a.O]) \mid a[d[out\, d.out\, e]] \mid \overline{open}\, d] \mid e[\overline{out}\, c] \\ &\xrightarrow{\tau} c[(\widetilde{n})(B \mid d[Q' \mid O]) \mid d[out\, e] \mid \overline{open}\, d] \mid e[\overline{out}\, c] \\ &\xrightarrow{\tau} c[(\widetilde{n})(B \mid d[Q' \mid O]) \mid \overline{open}\, d] \\ &\Longrightarrow c[Q'' \mid \overline{open}\, d] \end{aligned}$$

It then follows that $Q \Longrightarrow\overset{[out\, a]}{\longrightarrow} (\widetilde{n})\langle B\rangle Q'$ and

$$(\widetilde{n})(B \mid d[Q' \mid O]) \Longrightarrow Q'' \ \mathcal{R}\ (\widetilde{m})(A \mid d[P' \mid O])$$

– $\lambda = a[in\, b]$ and $P \overset{a[in\, b]}{\longrightarrow} (\widetilde{m})\langle A\rangle P'$. For process O and fresh e, let $C[\_]$ be the context

$$c[\_ \mid b[\overline{in}\, a.\overline{out}\, d.O] \mid d[d[out\, b.out\, e]] \mid \overline{open}\, d] \mid e[\overline{out}\, c]$$

Now clearly the reduction sequence

$$\begin{aligned} C[P] &\xrightarrow{\tau} c[(\widetilde{m})(P' \mid b[A \mid \overline{out}\, d.O]) \mid d[d[out\, b.out\, e]] \mid \overline{open}\, d] \mid e[\overline{out}\, c] \\ &\xrightarrow{\tau} c[(\widetilde{m})(P' \mid b[A \mid O]) \mid d[out\, e] \mid \overline{open}\, d] \mid e[\overline{out}\, c] \\ &\xrightarrow{\tau} c[(\widetilde{m})(P' \mid b[A \mid O]) \mid \overline{open}\, d] \end{aligned}$$

must be matched up by $C[Q] \overset{\tau}{\Longrightarrow} c[Q'' \mid \overline{open}\, d]$, which can be rearranged as follows:

$$\begin{aligned} C[Q] &\Longrightarrow C[Q_1] \\ &\xrightarrow{\tau} c[(\widetilde{n})(Q' \mid b[B \mid \overline{out}\, d.O]) \mid d[d[out\, b.out\, e]] \mid \overline{open}\, d] \mid e[\overline{out}\, c] \\ &\xrightarrow{\tau} c[(\widetilde{n})(Q' \mid b[B \mid O]) \mid d[out\, e] \mid \overline{open}\, d] \mid e[\overline{out}\, c] \\ &\xrightarrow{\tau} c[(\widetilde{n})(Q' \mid b[B \mid O]) \mid \overline{open}\, d] \\ &\Longrightarrow c[Q'' \mid \overline{open}\, d] \end{aligned}$$

It should be clear that $Q \Longrightarrow\overset{a[in\, b]}{\longrightarrow} (\widetilde{n})\langle B\rangle Q'$ and

$$(\widetilde{n})(Q' \mid b[B \mid O]) \Longrightarrow Q''\ \mathcal{R}\ (\widetilde{m})(P' \mid b[A \mid O])$$

– $\lambda = a[\overline{open}\, b]$ and $P \overset{a[\overline{open}\, b]}{\longrightarrow} (\widetilde{m})\langle M\rangle P'$. For process O and fresh e, let the context $C[\_]$ be $C'[\_ \mid b[open\, a.\overline{out}\, d.O]]$ where $C'[\_]$ is

$$c[\_ \mid d[d[out\, b.out\, e]] \mid \overline{open}\, d] \mid e[\overline{out}\, c]$$

The rest of the argument is completely the same as in the previous case.

– $\lambda = a[open\, b]$ and $P \overset{a[open\, b]}{\longrightarrow} C_1[\_]$. For process N and fresh e, let the context $C[\_]$ be

$$c[\_ \mid b[\overline{open}\, a.\overline{out}\, d.N] \mid d[d[out\, a.out\, e]] \mid \overline{open}\, d] \mid e[\overline{out}\, c]$$

Then

$$\begin{aligned} C[P] &\xrightarrow{\tau} c[C_1[\overline{out}\, d.N] \mid d[d[out\, a.out\, e]] \mid \overline{open}\, d] \mid e[\overline{out}\, c] \\ &\xrightarrow{\tau} c[C_1[N] \mid d[out\, e] \mid \overline{open}\, d] \mid e[\overline{out}\, c] \\ &\xrightarrow{\tau} c[C_1[N] \mid \overline{open}\, d] \end{aligned}$$

which is simulated by $C[Q]$ in the following manner

$$C[Q] \overset{\tau}{\Longrightarrow} c[C_2[N] \mid \overline{open}\, d] \overset{\tau}{\Longrightarrow} c[Q' \mid \overline{open}\, d]$$

Thus $Q \Longrightarrow\overset{a[open\, b]}{\longrightarrow} C_2[\_]$ and $C_2[N] \Longrightarrow Q'\mathcal{R}C_1[N]$.

– $\lambda = a[\overline{in}\, b]$ and $P \overset{a[\overline{in}\, b]}{\longrightarrow} C_1[\_]$. For process N and fresh names e, f, g, let the context $C[\_]$ be defined by

$$c[\_ \mid b[in\, a.(N \mid out\, f) \mid \overline{open}\, e.out\, e] \mid O]$$

where O is $e[open\, b \mid out\, g] \mid \overline{open}\, d$. Then clearly

$$C[P] \xrightarrow{\tau} c[C_1[b[N \mid out\, f \mid \overline{open}\, e.out\, e]] \mid O]$$

By Lemma 1, $C_1[\_]$ is an ambient context at a. The ambient $b[N \mid out\, f \mid \overline{open} e.out\, e]$ might get out of a. When it is out of a it could interact with the ambient $e[open\, b \mid out\, g]$. Thus we have the following important observation:

Either $c[C_1[b[N \mid out\, f \mid \overline{open}\, e.out\, e]] \mid O]$ is observable at both $c[[out\, e]]$ and $c[[out\, f]]$, or it is observable at neither $c[[out\, e]]$ nor $c[[out\, f]]$.

Suppose that the above action of $C[P]$ is simulated by

$$C[Q] \Longrightarrow A$$

It is impossible for A to take the form $C[Q_1]$ since one would then have the following reduction

$$A \xrightarrow{\tau} c[Q_1 \mid e[in\, a.(N \mid out\, f) \mid out\, e \mid out\, g] \mid \overline{open}\, d] \tag{24}$$

Here the component $in\, a.(N \mid out\, f)$ gets stuck since e does not appear in Q_1. It should be clear that

$$c[Q_1 \mid e[in\, a.(N \mid out\, f) \mid out\, e \mid out\, g] \mid \overline{open}\, d]$$

is observable at $c[[out\, e]]$. It is however *not* observable at $c[[out\, f]]$. It follows that the reduction (24) could not be matched up by any descendants of $c[C_1[b[N \mid out\, f \mid \overline{open}\, e.out\, e]] \mid O]$. Based upon the above analysis one concludes that the simulation must take the following form for some contexts $C_2[\_], C_3[\_]$:

$$\begin{aligned} C[Q] &\Longrightarrow C[Q_1] \\ &\xrightarrow{\tau} c[C_2[b[N \mid out\, f \mid \overline{open}\, e.out\, e]] \mid O] \\ &\Longrightarrow c[C_3[out\, f \mid \overline{open}\, e.out\, e] \mid O] \end{aligned}$$

By composing with the context $\_ \mid g[\overline{out}\, c] \mid b[\overline{open}\, e]$, it follows easily from Lemma 3 and the equivalence of the ambients $c[C_1[b[N \mid out\, f \mid \overline{open}\, e.out\, e]] \mid O]$ and $c[C_3[out\, f \mid \overline{open}\, e.out\, e] \mid O]$ that

$$c[C_1[b[N \mid out\, f \mid \overline{open}\, e.out\, e]] \mid \overline{open}\, d] \tag{25}$$

is observable bisimilar to

$$c[C_3[out\, f \mid \overline{open}\, e.out\, e] \mid \overline{open}\, d] \tag{26}$$

By applying the localization operator (e) and (f) to (25) and (26) one immediately sees that the ambient $c[C_1[b[N]] \mid \overline{open}\, d]$ is observable bisimilar to the ambient $c[C_3[\mathbf{0}] \mid \overline{open}\, d]$. We can now conclude that $Q \Longrightarrow\xrightarrow{a[\overline{in}\, b]} C_2[\_]$ and

$$C_2[b[N]] \Longrightarrow C_3[\mathbf{0}]\ \mathcal{R}\ C_1[b[N]]$$

By using the closure property of $\approx_{obs}$, it follows that $P \approx_{obs} Q$ implies $P \approx Q$. The implication of the other direction is straightforward by definition. □

Theorem 6.4 is significant not only in that it supports the bisimulation equivalence, but also in that its proof implies a property of local observability.

Theorem 6.5 (Local observability) *$P \approx Q$ if and only if $C[P] \approx C[Q]$ for every ambient context $C[\_]$.*

Two processes are bisimilar if and only if they induce bisimilar ambient actions. In other words, the equivalence of two non-ambients is strong enough to be closed under ambient contexts, but weak enough to be deducible from the equivalence on ambients. The local observability is a strong support to Definition 4.1.

In MA the local observability is a property too simple to be stated. As a matter of fact in MA a very strong property is forced upon us by the global open operator:

$a[P]$ is equivalent to $b[Q]$ if and only if $a = b$ and P is equivalent to Q.

The above property is not at all a merit. It says that one can not determine the equivalence of two MA ambients by comparing their observable actions. One has to see through the ambients as it were. For instance $a[b[\mathbf{0}]]$ and $a[c[\mathbf{0}]]$ are not equivalent even though neither can perform any observable actions. In the spirit of the observational theory, MA is best seen as a sub-calculus of SA using the embedding mentioned in Sect. 1.

In FA, $a[P] \approx a[Q]$ does not necessarily imply $P \approx Q$ and it may well be that $a[P] \approx b[Q]$ for $a \neq b$.

7 Barbed bisimulation

By exploring the labeled transition system, one could define in our framework the barbedness as follows:

Definition 7.1 P is barbed at a, notation $P\Downarrow a$, if $P \Longrightarrow\xrightarrow{a[\kappa]} U$ for some κ and U or $P \Longrightarrow\xrightarrow{a[[out\, b]]} V$ for some b and V. A relation $\mathcal{R}$ is barbed if $P\mathcal{R}Q$ implies $P\Downarrow a \Leftrightarrow Q\Downarrow a$ for every $a \in \mathcal{N}$.

The barbed bisimulations are then defined in the standard manner.

Definition 7.2 A symmetric relation $\mathcal{R}$ on processes is a barbed bisimulation if the following properties hold:

(i) It is barbed and is closed under context;
(ii) If $Q\mathcal{R}P \xrightarrow{\tau} P'$ then $Q \Longrightarrow Q'\mathcal{R}P'$ for some Q'.

The barbed bisimilarity $\approx_{brb}$ is the largest barbed bisimulation.

An observable relation is a barbed relation. It follows then that $\approx_{obs}\subseteq\approx_{brb}$. We will render the studies of the barbed bisimilarity unnecessary by proving that the reverse inclusion also holds. For that purpose we introduce a weaker form of the observable bisimilarity.

Definition 7.3 A relation $\mathcal{R}$ is weakly observable if $P\mathcal{R}Q \Rightarrow (P\Downarrow \Leftrightarrow Q\Downarrow)$.

Definition 7.4 A symmetric binary relation $\mathcal{R}$ is a weak observable bisimulation if the following properties hold:

(i) It is weakly observable and is closed under context;
(ii) If $Q\mathcal{R}P \xrightarrow{\tau} P'$ then $Q \Longrightarrow Q'\mathcal{R}P'$ for some Q'.

The weak observable bisimilarity $\approx_{wob}$ is the largest weak observable bisimulation.

The barbed bisimilarity lies between the observable bisimilarity and the weak observable bisimilarity.

Lemma 6 $\approx_{obs}\subseteq\approx_{brb}\subseteq\approx_{wob}$.

Proof Suppose $P \approx_{brb} Q$ and $P{\Downarrow}\lambda$. If λ is $a[\kappa]$ or $a[[out\, b]]$ then $P{\Downarrow}a$. If λ is κ or $[out\, a]$ then $c[P]{\Downarrow}c$. It follows from $P \approx_{brb} Q$ and $c[P] \approx_{brb} c[Q]$ that either $Q{\Downarrow}a$ or $c[Q]{\Downarrow}c$. Hence $Q{\Downarrow}\lambda'$ for some λ'. So $\approx_{brb}$ is a weak observable bisimulation. □

The lemma is not just formally interesting. For us it suggests that more attention should be paid to $\approx_{obs}$ and $\approx_{wob}$ than to $\approx_{brb}$. The equivalence $\approx_{obs}$ makes very good sense in the observation theory. The relation $\approx_{wob}$ looks much weaker. However we have the following result.

Theorem 7.5 *$\approx_{wob}$ is the same as $\approx_{obs}$.*

Proof Suppose $P \approx_{wob} Q$ and $P \Longrightarrow\xrightarrow{\lambda} U$. Let $\widetilde{x}$ be the free names in $P \mid Q$ and let d, e be fresh. The idea of the proof is to find a context $C[\_]$ of the form $(\widetilde{x})(\_ \mid O)$ such that the following conditions are met:

- $C[P] \Downarrow d[in\, e]$ and $C[Q] \Downarrow d[in\, e]$.
- $C[P] \Longrightarrow P' \not\Downarrow$ for some P' where P has performed the λ action.

It follows from these two conditions and $C[P] \approx_{wob} C[Q]$ that $C[Q] \Longrightarrow Q' \not\Downarrow$ for some Q', from which one infers that Q must be able to perform a λ action. The proof is carried out by case analysis:

- $\lambda = \kappa$. Notice that $a[P] \approx_{wob} a[Q]$ by definition and $a[P] \Longrightarrow\xrightarrow{a[\kappa]} U'$ for some U'. So this case is subsumed by the following cases.
- $\lambda = a[\overline{out}\, b]$. Let $C[\_]$ be the context defined by $(\widetilde{x})(\_ \mid b[e[out\, a.\overline{in}\, d]] \mid d[in\, e])$. Then the reductions $C[P] \overset{\tau}{\Longrightarrow} (\widetilde{x})(U \mid e[\overline{in}\, d] \mid d[in\, e]) \xrightarrow{\tau} (\widetilde{x})U$ must be simulated by $C[Q] \overset{\tau}{\Longrightarrow} (\widetilde{x})V$ for some V such that $Q \overset{a[\overline{out}\, b]}{\Longrightarrow} V$.
- $\lambda = a[[out\, b]]$. Let $C[\_]$ be the context $(\widetilde{x})(\_ \mid b[\overline{out}\, a.\overline{in}\, d] \mid d[in\, b \mid in\, e])$. Observe that $C[P] \overset{\tau}{\Longrightarrow} (\widetilde{x})(U \mid b[d[in\, e]]) \not\Downarrow$. Thus
$$(\widetilde{x})(Q \mid b[\overline{out}\, a.\overline{in}\, d] \mid d[in\, b \mid in\, e]) \overset{\tau}{\Longrightarrow} Q' \not\Downarrow$$
for some Q'. It is then clear that $Q \Downarrow a[[out\, b]]$.
- $\lambda = [out\, b]$. This is subsumed by the previous case.
- $\lambda = a[in\, b]$. Use the context $C[\_]$ defined below $(\widetilde{x})(\_ \mid b[\overline{in}\, a.\overline{in}\, d] \mid d[in\, b \mid in\, e])$.
- $\lambda = a[\overline{open}\, b]$. Use the context $C[\_]$ defined below
$$(\widetilde{x})(\_ \mid b[open\, a.open\, d] \mid d[\overline{open}\, b \mid in\, e])$$
- $\lambda = a[\overline{in}\, b]$. Use the context $C[\_]$ defined below
$$(\widetilde{x})(\_ \mid b[in\, a.out\, e] \mid e[\overline{out}\, a.\overline{in}\, d] \mid d[in\, e])$$
- $\lambda = a[open\, b]$. Use the context $C[\_]$ defined below
$$(\widetilde{x})(\_ \mid b[\overline{open}\, a.open\, d] \mid d[\overline{open}\, a \mid in\, e])$$

We conclude that $\approx_{wob}$ is an observable bisimulation. □

One interpretation of Theorem 7.5 is that there is very little overloading of semantics in the action labels. If two actions are different in syntax, they are different in semantics. In other words distinct actions have distinct impacts on the environments. If P can perform λ while Q can not, then they would have different impacts on the environments and therefore can be distinguished by the environments.

Theorem 7.5 adds additional weight to the labeled transition semantics.

8 Communication

MA was proposed as a model for mobility. As a computational model MA is Turing complete. From a programming point of view, the capacity to express the Turing computability is one thing, the ability to support particular computing structures or even programming methodologies is another issue. In MA all computations are interpreted in terms of ambient mobility. This is fine if one thinks of MA as a basic model for mobility. A different attitude is to regard MA as providing an open framework to support the computing mobility of all shapes and forms. The open-ended view is what a programming practitioner tends to adopt.

A well studied programming paradigm is the functional one. Both the λ-terms and the π-processes are functional programs reincarnated in different frameworks. Cardelli and Gordon have looked at the functional applications, or communications, in MA. They introduced two new forms of processes. One is $(x).P$, which can instantiate the bound name x by a received name, say y, and then evolves as $P\{y/x\}$. Another is $\langle x\rangle$ that may release the name x to whoever wants it before terminating. Computations of these new processes are basically the functional applications in a concurrent fashion. The reduction rule is the following:

$$(x).P \mid \langle y\rangle \longrightarrow P\{y/x\} \tag{27}$$

where $P\{y/x\}$ is obtained from P by replacing x by y throughout P. One criticism to (27) is that there is no way to tie up the two interacting parties. In the process $(x).P \mid \langle y\rangle \mid \langle z\rangle$ for example it is completely nondeterministic which name $(x).P$ is going to receive. In λ-calculus the terms M, N in MN are tightly coupled. In π-calculus one could specify a particular name to channel the interaction as in $(a)(a(x).P \mid \overline{a}y.Q)$. In MA the only control over communications is that two communicating processes must be in one ambient.

In this section we study the functional computation in the framework of FA. We shall stick to the three principles advocated in Sect. 1. The reduction defined in (27) has to be rejected since the interactions are not between two ambients. So we abandon the view that communications are local actions. As alternatives we propose two new capabilities that induce the interactions of ambients.

- The input capability $a(x)$ introduces processes of the form $a(x).P$. This is the process that is able to receive a name, say y, *from* ambient a, instantiates the bound name x by y, and then proceeds as $P\{y/x\}$.
- The output capability $\overline{a}x$ introduces processes of the form $\overline{a}x.P$. The process $\overline{a}x.P$ is capable of releasing the name x *to* ambient a and then continues as P.

Like the other capability operators, the interactional behaviors of the new capability operators can only be invoked from within ambients. The functional interactions they define are as follows:

$$a[L \mid b(x).P] \mid b[N \mid \overline{a}y.Q] \xrightarrow{\tau} a[L \mid P\{y/x\}] \mid b[N \mid Q]$$
$$a[b(x).P] \mid b[(y)\overline{a}y.Q] \xrightarrow{\tau} (y)(a[P\{y/x\}] \mid b[Q])$$

The syntax and the semantics of the functional aspects of FA are prescribed in Fig. 2. The extended language will be called functional FA. In the rules the symbol α ranges over the set $\mathcal{N} \cup \{\overline{a} \mid a \in \mathcal{N}\}$. The name x in $a(x).P$ is bound. The input operator $a(x)$. introduces *dummy* names, whereas the restriction operator (x) introduces *local*

Process with Communication

$P := \ldots \mid a(x).P \mid \overline{a}x.P$

Operational Semantics of Communication

$$\frac{}{a(x).P \xrightarrow{ay} P\{y/x\}} \qquad \frac{}{\overline{a}x.P \xrightarrow{\overline{a}x} P} \qquad \frac{P \xrightarrow{\alpha x} P'}{b[P] \xrightarrow{b[\alpha x]} b[P']} \qquad \frac{P \xrightarrow{\overline{a}(x)} P'}{b[P] \xrightarrow{(x)b[\overline{a}x]} b[P']}$$

$$\frac{P \xrightarrow{\overline{a}x} P'}{(x)P \xrightarrow{\overline{a}(x)} P'} \qquad \frac{P \xrightarrow{b[\overline{a}x]} P'}{(x)P \xrightarrow{(x)b[\overline{a}x]} P'} \qquad \frac{P \xrightarrow{b[ax]} P' \quad Q \xrightarrow{a[\overline{b}x]} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\frac{P \xrightarrow{b[ax]} P' \quad Q \xrightarrow{(x)a[\overline{b}x]} Q'}{P \mid Q \xrightarrow{\tau} (x)(P' \mid Q')} x \notin fn(P)$$

Fig. 2 Functional FA

names. Both are bound names. The notation $bn(P)$ stands for the set of bound names in P; and $fn(P)$ denotes the set of free, or non-bound, names in P. We shall abbreviate $(x)\overline{a}x.P$ to $\overline{a}(x).P$. When $x \notin fn(P)$, we write $\alpha.P$ for $\alpha(x).P$.

The labeled transition system introduces six new kinds of labels. The intended meanings of these labels are as follows:

- "ax" reads "import the name x from a".
- "$\overline{a}x$" reads "export the name x to a".
- "$\overline{a}(x)$" reads "export the bound name x to a".
- "$a[bx]$" reads "ambient a imports the name x from ambient b".
- "$a[\overline{b}x]$" reads "ambient a exports the name x to ambient b".
- "$(x)a[\overline{b}x]$" reads "ambient a exports the bound name x to ambient b".

The actions denoted by these new labels will be called functional actions. Accordingly the new capabilities will be called functional capabilities.

The functional capability provides a simple mechanism to synchronize the concurrent computations of ambients in the following manner:

$$a[b.P] \mid b[\overline{a}.Q] \xrightarrow{\tau} a[P] \mid b[Q]$$

A special case of the above synchronization is

$$a[a.P] \mid a[\overline{a}.Q] \xrightarrow{\tau} a[P] \mid a[Q]$$

The ambient $a[a.P]$ is almost like the CCS process $a.P$. It will be clear in the next section that CCS can be embedded into the functional FA.

8.1 More examples

Three examples are given in this subsection to demonstrate the expressiveness of the communications between ambients. A more fundamental example illustrating the expressive power of the functional FA will be given in the next section.

Example 11 Using the functional capabilities, the transportation of resources from one ambient to another ambient can be made more secure as in the following example:

$$\begin{aligned}
& a[N \mid (v)\overline{b}v.v[\mathit{out}\, b.\overline{\mathit{open}}\, b.P]] \mid b[a(x).\overline{\mathit{out}}\, a.\mathit{open}\, x.Q] \\
& \xrightarrow{\tau} (v)(a[N \mid v[\mathit{out}\, b.\overline{\mathit{open}}\, b.P]] \mid b[\overline{\mathit{out}}\, a.\mathit{open}\, v.Q]) \\
& \xrightarrow{\tau} (v)(v[\overline{\mathit{open}}\, b.P] \mid a[N] \mid b[\mathit{open}\, v.Q]) \\
& \xrightarrow{\tau} (v)(a[N] \mid b[P \mid Q])
\end{aligned}$$

The communication of a local name makes sure that an ambient will not choose a wrong target in the middle of the movement from one ambient to another. □

Example 12 Using the functional capability one could give a much better modeling of the firewall protocol defined in Example 9. An agent intends to upload a piece of resource to a host. The host is protected by a firewall. The two are defined as follows:

$$\begin{aligned}
\mathit{Firewall} &\overset{\text{def}}{=} (\mathit{fw})(b)\mathit{fw}[K_0 \mid b(y).\mathit{open}\ y \mid F] \\
\mathit{Agent} &\overset{\text{def}}{=} (k_{ag})(c)\mathit{ag}[K_1 \mid V \mid c(f).\overline{\mathit{open}}\ f \mid r[R]]
\end{aligned}$$

where

$$\begin{aligned}
K_0 &\overset{\text{def}}{=} k_0[\mathit{out}.k_1(k).k[\mathit{out}.k_1(n).k_1(z).\overline{n}z.\overline{\mathit{out}}\ n.k_{ag}(c).X]] \\
X &\overset{\text{def}}{=} b[\mathit{out}.\overline{\mathit{fw}}n] \mid c[\mathit{out}.\overline{n}\mathit{fw}] \\
K_1 &\overset{\text{def}}{=} k_1[\mathit{out}.\overline{k_0}(k).\overline{k}\mathit{ag}.\overline{k}k_{ag}.c[\mathit{out}.\overline{\mathit{ag}}k]] \\
V &\overset{\text{def}}{=} c(k).k(z).(v)(k_{ag}[\overline{v}.\mathit{out}\ k.\overline{k}c] \mid v[z])
\end{aligned}$$

Let *Sys* be the system *Firewall* | *Agent*. The system *Sys* can engage in a sequence of talks between its two subsystems:

$$\begin{aligned}
\mathit{Sys} & \xrightarrow{\tau}{}^{*} (\mathit{fw})(b)(\mathit{fw}[\ldots] \mid k_0[k_1(k).\ldots]) \mid (k_{ag})(c)(\mathit{ag}[\ldots] \mid k_1[\overline{k_0}(k).\ldots]) \\
& \xrightarrow{\tau} (k)((\mathit{fw})(b)(\mathit{fw}[\ldots] \mid k_0[k[\mathit{out}.\ldots]]) \mid (k_{ag})(c)(\mathit{ag}[\ldots] \mid k_1[\overline{k}\mathit{ag}.\ldots])) \\
& \xrightarrow{\tau} (k)((\mathit{fw})(b)(\mathit{fw}[\ldots] \mid k[k_1(n).\ldots]) \mid (k_{ag})(c)(\mathit{ag}[\ldots] \mid k_1[\overline{k}\mathit{ag}.\ldots])) \\
& \xrightarrow{\tau} (k)((\mathit{fw})(b)(\mathit{fw}[\ldots] \mid k[k_1(z).\ldots]) \mid (k_{ag})(c)(\mathit{ag}[\ldots] \mid k_1[\overline{k}k_{ag}.\ldots])) \\
& \xrightarrow{\tau} (k)(k_{ag})((\mathit{fw})(b)(\mathit{fw}[\ldots] \mid k[\overline{\mathit{ag}}k_{ag}.\ldots]) \mid (c)(\mathit{ag}[\ldots] \mid k_1[c[\mathit{out}.\overline{\mathit{ag}}k]])) \\
& \xrightarrow{\tau} (k)(k_{ag})((\mathit{fw})(b)(\mathit{fw}[\ldots] \mid k[\overline{\mathit{ag}}k_{ag}.\ldots]) \mid (c)(\mathit{ag}[c(k).\ldots] \mid c[\overline{\mathit{ag}}k])) \\
& \xrightarrow{\tau} (k)(k_{ag})((\mathit{fw})(b)(\mathit{fw}[\ldots] \mid k[\overline{\mathit{ag}}k_{ag}.\ldots]) \mid (c)\mathit{ag}[k(z).\ldots]) \\
& \xrightarrow{\tau} (k)(k_{ag})(\ldots \mid (c)(\mathit{ag}[(v)(k_{ag}[\overline{v}.\mathit{out}\ k.\overline{k}c] \mid v[k_{ag}]) \mid \ldots]) \\
& \xrightarrow{\tau} (k)(k_{ag})((\mathit{fw})(b)(\mathit{fw}[\ldots] \mid k[\overline{\mathit{out}}\ \mathit{ag}.\ldots]) \mid (c)(\mathit{ag}[k_{ag}[\mathit{out}\ k.\overline{k}c] \mid \ldots]) \\
& \xrightarrow{\tau} (k)(k_{ag})((\mathit{fw})(b)(\mathit{fw}[\ldots] \mid k[k_{ag}(c).X]) \mid (c)(k_{ag}[\overline{k}c] \mid \mathit{ag}[\ldots])) \\
& \xrightarrow{\tau} (k)(\mathit{fw})(b)(c)(\mathit{fw}[\ldots] \mid k[b[\mathit{out}.\overline{\mathit{fw}}\mathit{ag}] \mid c[\mathit{out}.\overline{\mathit{ag}}\mathit{fw}]] \mid \mathit{ag}[\ldots]) \\
& \xrightarrow{\tau}{}^{*} (\mathit{fw})(b)(c)(\mathit{fw}[b(y).\mathit{open}\ y \mid F] \mid b[\overline{\mathit{fw}}\mathit{ag}] \mid c[\overline{\mathit{ag}}\mathit{fw}] \mid \mathit{ag}[c(f).\overline{\mathit{open}}\ f \mid r[R]]) \\
& \xrightarrow{\tau}{}^{*} (\mathit{fw})(\mathit{fw}[\mathit{open}\ \mathit{ag} \mid F] \mid \mathit{ag}[\overline{\mathit{open}}\ \mathit{fw}.r[R]]) \\
& \xrightarrow{\tau} (\mathit{fw})\mathit{fw}[r[R] \mid F]
\end{aligned}$$

For the agent to successfully upload the resource $r[R]$ to the anonymous ambient, it has to pass a sequence of authentications:

1. The protected ambient *fw* releases an *fw*-envoy with the key k_0; the agent *ag* releases an *ag*-envoy with the key k_1; The pair k_0, k_1 of keys are used to initiate an negotiation;
2. The envoys, who know each other's keys, establish a link with a secret key k;
3. The *fw*-envoy starts to use the secret key k for further communication with the *ag*-envoy;
4. The *fw*-envoy receives the name of the agent from the *ag*-envoy;
5. The *fw*-envoy receives a secret key k_{ag} of the agent from the *ag*-envoy;
6. The *ag*-envoy discards the key k_1;
7. The agent *ag* receives the secret key k;
8. The *fw*-envoy sends the secret key k_{ag}, received from the *ag*-envoy, back to the agent;
9. The agent verifies the received key k_{ag};
10. After the successful verification, the agent signals this message to the *fw*-envoy;
11. Then the personal key of the agent is used to pass the password of the agent to the *fw*-envoy;
12. The *fw*-envoy releases two subenvoys and dissolves itself;
13. The subenvoys establish a direct link between the protected host and the agent;
14. Finally the transmission of the resource can happen.

The purpose of this example is to explain how communications between ambients can be explored to specify protocols. □

Example 13 (Signal transduction) Signal transduction at the cellular level refers to the movement of signals from outside the cell to inside. The movement of signals can be simple. Upon ligand interaction, a cell would open a channel to allow signals to be passed in the form of small ion movement, either into or out of the cell. These ion movements result in changes in the electrical potential of the cells that, in turn, propagates the signal along the cell. The following describes a situation when an ion, as a signal, enters a cell:

$$\begin{aligned} Env \mid Cell \mid Ion &\xrightarrow{\tau} env[P_1] \mid cell[\overline{in}\ ion.P_2 \mid ligand[out\ env' \mid P_3] \mid P_4] \mid ion[in\ cell \mid P_5] \\ &\xrightarrow{\tau} env[P_1] \mid cell[P_2 \mid ligand[out\ env' \mid P_3] \mid P_4 \mid ion[P_5]] \end{aligned}$$

where

$$\begin{aligned} Env &\overset{\text{def}}{=} env[\overline{cell}lingand.P_1] \\ Cell &\overset{\text{def}}{=} cell[env(ligand).(\overline{in}\ ion.P_2 \mid ligand[out\ env' \mid P_3]) \mid P_4] \\ Ion &\overset{\text{def}}{=} ion[in\ cell \mid P_5] \end{aligned}$$

This example makes use of a global communication between two ambients. □

8.2 Observational theory

In the rest of this section we shall focus on the observational theory of FA. Let κ range over the extended set of capabilities and λ over the extended set of labels. If we simply modify Definition 4.1 by replacing the third clause by the following requirement:

If $P \xrightarrow{\lambda} P'$ and λ is a label in the following set

$$\{a[\overline{out}\,b], a[[out\,b]], a[bx], a[\overline{b}x], (x)a[\overline{b}x] \mid a, b \in \mathcal{N}\}$$

then $Q \overset{\lambda}{\Longrightarrow} Q'\mathcal{R}P'$ for some Q'.

We would get an equivalence relation for the functional ambients. But it fails to be a congruence. This is obvious for people familiar with the π-calculus. The process $(a)(a[\overline{in}\,x] \mid b[in\,a.out\,x])$ is bisimilar to $\mathbf{0}$ in the sense of Definition 4.1. However $d(x).(a)(a[\overline{in}\,x] \mid b[in\,a.out\,x])$ is not bisimilar to $d(x).\mathbf{0}$. In the literature on π-calculus several bisimulation congruence relations have been proposed. In our opinion the most attractive bisimulation congruence is the open bisimilarity of Sangiorgi [37]. For some special π-processes the open bisimilarity is too strong to be reasonable from an observational viewpoint. A rectified version, the quasi open bisimilarity, is proposed by Sangiorgi and Walker [40] and further studied by Fu [15]. The reader is advised to consult [40,15] for more information. Appendix C summarizes some relevant definitions and results on the quasi open bisimulations of the π-calculus.

Before defining bisimulations for the functional FA, some notations have to be fixed. A substitution σ of name, or simply substitution, is a map from $\mathcal{N}$ to $\mathcal{N}$ such that the set $\{x \mid x \neq \sigma(x)\}$ is finite. The set of substitutions will be ranged over by σ. We write $P\sigma$ for the result of applying σ to P. The finite subset relation is denoted by $\subseteq_f$.

A well known difficulty for bisimulations on the π-processes is how to define the simulations for the bound output actions. In functional FA, the question is how to simulate $P \xrightarrow{\overline{a}(x)} P'$ and $P \xrightarrow{(x)a[\overline{b}x]} P'$. The caution must be paid not to confuse the name x in P' with any other free names. Now if bisimulations are closed under all environments, they must also be closed under substitution of names. That suggests that some mechanism should be designed that only admits substitutions that respect local names. The motto is that we treat the local names as if they were free names, but we have to respect them. This is why we need the following definition.

Definition 8.1 A substitution σ respects $\widetilde{x}$ if $\forall x \in \widetilde{x}.x\sigma = x$ and $\forall y \notin \widetilde{x}.y\sigma \notin \widetilde{x}$.

Another important issue is how to make sure that bisimulations for the functional FA give rise to a congruence relation. A congruence is a relation closed under all contexts. Now for functional FA the contexts are redefined to take into account the functional capability:

- A context defined in Definition 2.1 is an open context;
- If $C[\_]$ is an open context then $a(x).C[\_]$ and $\overline{a}x.C[\_]$ are open contexts.

The observable bisimilarity, the barbed bismilarity and the weak observable bisimilarity for the functional FA are defined by replacing the phrase "context" in Definition 6.3, Definition 7.2 and Definition 7.4 respectively by the phrase "open context". We write $\approx^f_{obs}$, $\approx^f_{brb}$ and $\approx^f_{wob}$ for the three respective bisimilarities on the functional FA processes.

For the counterpart of Definition 4.1 in the functional FA one can not simply require that bisimulations are closed under contexts since that would defeat the definition. Closure under the functional capabilities implies the closure of substitutions of names. This brings us to Sangiorgi's open approach.

The above discussions lead us to combine Definition 4.1 with the definition of the quasi open bisimulations of the π-calculus for the purpose of defining a bisimulation equivalence for the functional FA.

Definition 8.2 A family of symmetric relations $\{\mathcal{R}^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f \mathcal{N}}$ on processes is an open bisimulation if whenever $P\mathcal{R}^{\widetilde{z}}Q$ and σ respects $\widetilde{z}$ then $P\sigma\mathcal{R}^{\widetilde{z}}Q\sigma$ and the followings hold:

1. If $P \xrightarrow{\tau} P'$ then $Q \Longrightarrow Q'\mathcal{R}^{\widetilde{z}}P'$ for some Q'.
2. If $P \xrightarrow{\kappa} P'$ and κ is not of the form $\overline{a}(x)$, then $Q \xRightarrow{\kappa} Q'\mathcal{R}^{\widetilde{z}}P'$ for some Q'.
3. If $P \xrightarrow{\overline{a}(x)} P'$ then $Q \xRightarrow{\overline{a}(x)} Q'\mathcal{R}^{\widetilde{z}x}P'$ for some Q'.
4. If $P \xrightarrow{\lambda} P'$ and λ is a label in the following set
$$\{a[\overline{out}\, b], a[[out\, b]], a[bx], a[\overline{b}x] \mid a, b \in \mathcal{N}\}$$
then $Q \xRightarrow{\lambda} Q'\mathcal{R}^{\widetilde{z}}P'$ for some Q'.
5. If $P \xrightarrow{(x)a[\overline{b}x]} P'$ then $Q \xRightarrow{(x)a[\overline{b}x]} Q'\mathcal{R}^{\widetilde{z}x}P'$ for some Q'.
6. If $P \xrightarrow{[out\, a]} (\widetilde{m})\langle A\rangle P'$ then for every name d and every process O satisfying $\{\widetilde{m}\}\cap fn(O) = \emptyset$ there exist some $\widetilde{n}, B, Q', Q''$ such that $Q \Longrightarrow\xrightarrow{[out\, a]} (\widetilde{n})\langle B\rangle Q'$ and $(\widetilde{n})(B \mid d[Q' \mid O]) \Longrightarrow Q''\ \mathcal{R}^{\widetilde{z}}\ (\widetilde{m})(A \mid d[P' \mid O])$.
7. If $P \xrightarrow{a[in\, b]} (\widetilde{m})\langle A\rangle P'$ then for every process O satisfying $\{\widetilde{m}\} \cap fn(O) = \emptyset$ there exist some $\widetilde{n}, B, Q', Q''$ such that $Q \Longrightarrow\xrightarrow{a[in\, b]} (\widetilde{n})\langle B\rangle Q'$ and $(\widetilde{n})(Q' \mid b[B \mid O]) \Longrightarrow Q''\ \mathcal{R}^{\widetilde{z}}\ (\widetilde{m})(P' \mid b[A \mid O])$.
8. If $P \xrightarrow{a[\overline{open}\, b]} (\widetilde{m})\langle M\rangle P'$ then for every process O satisfying $\{\widetilde{m}\} \cap fn(O) = \emptyset$ some $\widetilde{n}, N, Q', Q''$ exist such that $Q \Longrightarrow\xrightarrow{a[\overline{open}\, b]} (\widetilde{n})\langle N\rangle Q'$ and $(\widetilde{n})(Q' \mid b[N \mid O]) \Longrightarrow Q''\ \mathcal{R}^{\widetilde{z}}\ (\widetilde{m})(P' \mid b[M \mid O])$.
9. If $P \xrightarrow{a[\overline{in}\, b]} C[\_]$ then for every N satisfying $fn(N) \cap bn(C[\_]) = \emptyset$ there exist some $C'[\_]$, Q' such that $Q \Longrightarrow\xrightarrow{a[\overline{in}\, b]} C'[\_]$ and $C'[b[N]] \Longrightarrow Q'\ \mathcal{R}^{\widetilde{z}}\ C[b[N]]$.
10. If $P \xrightarrow{a[open\, b]} C[\_]$ then for every N satisfying $fn(N) \cap bn(C[\_]) = \emptyset$ there exist some $C'[\_]$, Q' such that $Q \Longrightarrow\xrightarrow{a[open\, b]} C'[\_]$ and $C'[N] \Longrightarrow Q'\ \mathcal{R}^{\widetilde{z}}\ C[N]$.

We write $\{\approx^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f \mathcal{N}}$ for the largest open bisimulation and refer to $\approx^{\widetilde{z}}$ as the open $\widetilde{z}$-bisimilarity. The open bisimilarity $\approx^f_{opn}$ is the open $\emptyset$-bisimilarity $\approx^{\emptyset}$.

In Definition 8.2 the clause 3 calls for a little explanation. Now $P\mathcal{R}^{\widetilde{z}}Q$ means that P and Q are equivalent under the assumption that $\widetilde{z}$ are distinct names that must be kept distinct from the other names. This is a reasonable assumption if $\widetilde{z}$ were local names that have been opened up by restricted output actions. After both P and Q open up the local name x, it must be kept distinct in further comparison since in reality environments can have absolutely no effects on x. That's why we require that $P'\mathcal{R}^{\widetilde{z}x}Q'$. The same intuition is behind clause 5.

The open bisimilarity is a congruent equivalence, the proof of which can be found in Appendix B.

Proposition 8.3 *The equivalence* $\approx^f_{opn}$ *is congruent.*

The observational theory of the functional FA calls for some technical lemmas, some of which are stated below. The proofs of these lemmas are straightforward applications of the Bisimulation lemma.

Lemma 7 *Suppose* $(x)(P \mid a[\overline{b}x]) \approx^{f}_{obs} (x)(Q \mid a[\overline{b}x])$ *for distinct names a and b. If either a or b is fresh then* $(x)(P \mid c[\overline{d}x]) \approx^{f}_{obs} (x)(Q \mid c[\overline{d}x])$ *for arbitrarily distinct fresh names* c, d.

Proof For fresh c, d, one has that

$$(x)(P \mid a[\overline{b}x]) \mid b[a(z).\overline{c}z] \mid c[b(z).\overline{d}z] \Longrightarrow (x)(P \mid c[\overline{d}x])$$

Now $(x)(P \mid c[\overline{d}x]) \approx^{f}_{obs} (x)(Q \mid c[\overline{d}x])$ follows by standard argument using Bisimulation lemma. □

Lemma 8 *If* $(x)(P \mid a[\overline{b}x]) \approx^{f}_{obs} (x)(Q \mid a[\overline{b}x])$ *for distinct fresh names a and b, then* $(x)P \approx^{f}_{obs} (x)Q$.

Proof Use the context $\_ \mid b[a(z)]$ to consume the ambient $a[\overline{b}x]$. □

Lemma 9 $(x)(y)(P \mid a[\overline{b}x] \mid c[\overline{d}x] \mid e[\overline{f}y])$ *and* $(x)(y)(Q \mid a[\overline{b}x] \mid c[\overline{d}y] \mid e[\overline{f}y])$ *are not bisimilar if* a, b, c, d, e, f *are fresh and pairwise distinct.*

Proof The context $\_ \mid d[c(z).\overline{f}z] \mid f[e(z).d(v).(\overline{z} \mid v.f)]$ is able to observe the difference between the two processes. □

Corollary 8.4 *If* a, b, c, d *are fresh and pairwise distinct then* $(x)(Q \mid a[\overline{b}x] \mid c[\overline{d}x])$ *is not bisimilar to* $(x)(y)(P \mid a[\overline{b}x] \mid c[\overline{d}y])$.

Proof The non-equivalence follows immediately from Lemmas 8 and 9. □

Corollary 8.5 $(x)(P \mid a[\overline{b}x]) \not\approx^{f}_{obs} Q \mid a[\overline{b}y]$ *if* a, b *are fresh and distinct.*

Proof Had $(x)(P \mid a[\overline{b}x]) \approx^{f}_{obs} Q \mid a[\overline{b}y]$ been held, then $(x)(y)(P \mid a[\overline{b}x] \mid c[\overline{d}y]) \approx^{f}_{obs} (y)(Q \mid a[\overline{b}y] \mid c[\overline{d}y])$ would have been held for fresh c, d, which would contradict to Corollary 8.4. □

The proof of the next result is omitted since it is similar to the proof of Lemma 5. We remark that the property is stable under substitution of name.

Lemma 10 *Suppose* $d, a_1, \ldots, a_n, b_1, \ldots, b_n$ *are fresh and pairwise distinct names. If the ambient* $(x_1, \ldots, x_n)(c[P \mid \overline{open}\, d] \mid a_1[\overline{b}_1 x_1] \mid \ldots \mid a_n[\overline{b}_n x_n])$ *is observable bisimilar to the ambient* $(x_1, \ldots, x_n)(c[Q \mid \overline{open}\, d] \mid a_1[\overline{b}_1 x_1] \mid \ldots \mid a_n[\overline{b}_n x_n])$*, then*

$$(x_1, \ldots, x_n)(C[P] \mid a_1[\overline{b}_1 x_1] \mid \ldots \mid a_n[\overline{b}_n x_n])$$

is observable bisimilar to the ambient $(x_1, \ldots, x_n)(C[Q] \mid a_1[\overline{b}_1 x_1] \mid \ldots \mid a_n[\overline{b}_n x_n])$ *for every ambient context* $C[\_]$ *that does not contain* $a_1, \ldots, a_n, b_1, \ldots, b_n$.

We are now ready to prove that the bisimilarities we introduced for FA also coincide in the functional FA. The following theorem states a special Local Observability.

Theorem 8.6 *Suppose $d, a_1, \ldots, a_n, b_1, \ldots, b_n$ are fresh names that are pairwise distinct. If $(x_1, \ldots, x_n)(c[P \,|\, \overline{open}\, d] \,|\, a_1[\overline{b}_1 x_1] \,|\, \ldots \,|\, a_n[\overline{b}_n x_n])$ is observable bisimilar to $(x_1, \ldots, x_n)(c[Q \,|\, \overline{open}\, d] \,|\, a_1[\overline{b}_1 x_1] \,|\, \ldots \,|\, a_n[\overline{b}_n x_n])$, then $P \approx^f_{opn} Q$.*

Proof Let $\{\mathcal{R}^{\widetilde{x}}\}_{\widetilde{x} \subseteq_f \mathcal{N}}$ be the family of relations where $\mathcal{R}^{\widetilde{x}}$, for $\widetilde{x} = x_1, \ldots, x_n$, is defined by the following relation

$$\left\{ (P, Q) \;\middle|\; \begin{array}{l} (\widetilde{x})(c[P \,|\, \overline{open}\, d] \,|\, a_1[\overline{b}_1 x_1] \,|\, \ldots \,|\, a_n[\overline{b}_n x_n]) \\ \qquad\qquad\qquad \approx^f_{obs} \\ (\widetilde{x})(c[Q \,|\, \overline{open}\, d] \,|\, a_1[\overline{b}_1 x_1] \,|\, \ldots \,|\, a_n[\overline{b}_n x_n]) \\ \text{where } d, a_1, \ldots, a_n, b_1, \ldots, b_n \text{ are fresh} \\ \text{and pairwise distinct, and } \widetilde{x} = x_1, \ldots, x_n \end{array} \right\}$$

Using Lemma 7 it is easy to show that the family $\{\mathcal{R}^{\widetilde{x}}\}_{\widetilde{x} \subseteq_f \mathcal{N}}$ is closed under substitutions in the sense that if $P \mathcal{R}^{\widetilde{x}} Q$ and σ respects $\widetilde{x}$ then $P\sigma \mathcal{R}^{\widetilde{x}} Q\sigma$. This observation reduces the complexity of the proof that $\{\mathcal{R}^{\widetilde{x}}\}_{\widetilde{x} \subseteq_f \mathcal{N}}$ is an open bisimulation. Now suppose $Q \,\mathcal{R}^{\widetilde{x}}\, P \xrightarrow{\lambda} P'$. To avoid long expressions, we introduce the following abbreviations:

$$A \stackrel{\text{def}}{=} (\widetilde{x})(c[P \,|\, \overline{open}\, d] \,|\, a_1[\overline{b}_1 x_1] \,|\, \ldots \,|\, a_n[\overline{b}_n x_n])$$
$$A' \stackrel{\text{def}}{=} (\widetilde{x})(c[P' \,|\, \overline{open}\, d] \,|\, a_1[\overline{b}_1 x_1] \,|\, \ldots \,|\, a_n[\overline{b}_n x_n])$$
$$B \stackrel{\text{def}}{=} (\widetilde{x})(c[Q \,|\, \overline{open}\, d] \,|\, a_1[\overline{b}_1 x_1] \,|\, \ldots \,|\, a_n[\overline{b}_n x_n])$$

It follows from Lemmas 7 and 10 and the fact $A \approx^f_{obs} B$ that $C[P] \mathcal{R}^{\widetilde{x}} C[Q]$ for every ambient context $C[\_]$. We will carry out a case analysis for the functional actions.

- $\lambda = ax$. This case is simple.
- $\lambda = \overline{a}(x)$. Then for fresh e, k, one has that

$$A \,|\, a[c(z).\overline{out}\, k.\overline{e}z] \,|\, k[k[out\, a]] \stackrel{\tau}{\Longrightarrow} (x)(A' \,|\, a[\overline{e}x])$$

  Thus $B \,|\, a[c(z).\overline{out}\, k.\overline{e}z] \,|\, k[k[out\, a]] \stackrel{\tau}{\Longrightarrow} N$ and $N \approx^f_{obs} (x)(A' \,|\, a[\overline{e}x])$ for some N. By Corollary 8.5, N can not be of the form $B' \,|\, a[\overline{e}v]$ for some free v. So N is of the form $(x)(B' \,|\, a[\overline{e}x])$ for some B'. It follows from Lemma 7 that

$$(x)(A' \,|\, a'[\overline{b'}x]) \approx^f_{obs} (x)(B' \,|\, a'[\overline{b'}x])$$

  for fresh a', b'. Thus $Q \stackrel{\overline{a}(x)}{\Longrightarrow} Q' \,\mathcal{R}^{\widetilde{x}x}\, P'$.
- $\lambda = \overline{a}x$ and $x \notin \widetilde{x}$. The proof is similar.
- $\lambda = \overline{a}x_i$. Then for fresh e, k, one has that

$$A \,|\, a[c(z).\overline{out}\, k.\overline{e}z] \,|\, k[k[out\, a]] \stackrel{\tau}{\Longrightarrow} (x_i)(A' \,|\, a[\overline{e}x_i])$$

  By Corollary 8.5 the reductions can not be simulated by

$$B \,|\, a[c(z).\overline{out}\, k.\overline{e}z] \,|\, k[k[out\, a]] \Longrightarrow L \,|\, a[\overline{e}w]$$

  for some free w. By Corollary 8.4 the reductions can not be simulated by

$$\begin{aligned} &B \,|\, a[c(z).\overline{out}\, k.\overline{e}z] \,|\, k[k[out\, a]] \\ &\Longrightarrow (y)(\widetilde{x})(c[Q_1 \,|\, \overline{open}\, d] \,|\, a_1[\overline{b}_1 x_1] \,|\, \ldots \,|\, a_n[\overline{b}_n x_n] \,|\, a[\overline{e}y]) \end{aligned}$$

for some local y. By Lemma 9 the reductions can not be simulated by

$$\begin{aligned} &B \mid a[c(z).\overline{out}\,k.\bar{e}z] \mid k[k[out\,a]] \\ &\Longrightarrow (\widetilde{x})(c[Q_2 \mid \overline{open}\,d] \mid a_1[\overline{b}_1x_1] \mid \ldots \mid a_n[\overline{b}_nx_n] \mid a[\overline{e}x_j]) \end{aligned}$$

for some $j \neq i$. It follows that the reductions can only be simulated by

$$\begin{aligned} &B \mid a[c(z).\overline{out}\,k.\bar{e}z] \mid k[k[out\,a]] \\ &\quad\Longrightarrow (\widetilde{x})(c[Q' \mid \overline{open}\,d] \mid a_1[\overline{b}_1x_1] \mid \ldots \mid a_n[\overline{b}_nx_n] \mid a[\overline{e}x_i]) \end{aligned}$$

for some Q'. It then follows from Lemmas 7 and 8 that $Q \overset{\overline{a}x_i}{\Longrightarrow} Q' \mathcal{R}^{\widetilde{x}} P'$.

- $\lambda = b[ax]$. Let $C[\_]$ be the context $(\widetilde{x})(C'[\_] \mid a_1[\overline{b}_1x_1] \mid \ldots \mid a_n[\overline{b}_nx_n]) \mid e[\overline{out}\,c]$, where $C'[\_]$ is $c[\_ \mid a[\overline{b}x.out\,e] \mid open\,d]$ and e is fresh. Now $C[P] \xrightarrow{\tau}\xrightarrow{\tau} A'$ must be simulated by $C[Q] \overset{\tau}{\Longrightarrow} (\widetilde{x})(c[Q' \mid \overline{open}\,d] \mid a_1[\overline{b}_1x_1] \mid \ldots \mid a_n[\overline{b}_nx_n])$ in such a way that $Q \overset{b[ax]}{\Longrightarrow} Q' \mathcal{R}^{\widetilde{x}} P'$.
- $\lambda = b[\overline{a}x]$ and $x \notin \widetilde{x}$. Let $C[\_]$ be defined by

$$(\widetilde{x})(C'[\_] \mid a_1[\overline{b}_1x_1] \mid \ldots \mid a_n[\overline{b}_nx_n]) \mid e[\overline{out}\,c.\overline{out}\,a \mid \overline{x}]$$

where $C'[\_]$ is $c[\_ \mid a[b(z).out\,e.z[out\,e.e]] \mid open\,d]$ and e is fresh. Now

$$\begin{aligned} C[P] &\xrightarrow{\tau} (\widetilde{x})(c[P' \mid a[out\,e.x[out\,e.e]] \mid open\,d] \mid \ldots \\ &\xrightarrow{\tau} (\widetilde{x})(c[P' \mid open\,d] \mid a[x[out\,e.e]] \mid \ldots) \mid e[\overline{out}\,a \mid \overline{x}] \\ &\xrightarrow{\tau} (\widetilde{x})(c[P' \mid open\,d] \mid x[e] \mid \ldots) \mid e[\overline{x}] \\ &\xrightarrow{\tau} (\widetilde{x})(c[P' \mid open\,d] \mid a_1[\overline{b}_1x_1] \mid \ldots \mid a_n[\overline{b}_nx_n]) \end{aligned}$$

must be simulated by $C[Q] \xrightarrow{\tau} (\widetilde{x})(c[Q' \mid open\,d] \mid a_1[\overline{b}_1x_1] \mid \ldots \mid a_n[\overline{b}_nx_n])$. Thus $Q \overset{b[\overline{a}x]}{\Longrightarrow} Q'\mathcal{R}^{\widetilde{x}}P'$.
- $\lambda = (y)b[\overline{a}y]$. Let the context $C[\_]$ be defined by

$$(\widetilde{x})(C'[\_] \mid a_1[\overline{b}_1x_1] \mid \ldots \mid a_n[\overline{b}_nx_n]) \mid e[\overline{out}\,c \mid \overline{out}\,a]$$

where $C'[\_]$ is $c[\_ \mid a[b(z).out\,e.k[out\,e.\overline{h}z]] \mid open\,d]$ and e, h, k are fresh. We have the following similar reductions:

$$\begin{aligned} C[P] &\xrightarrow{\tau} (\widetilde{x})(y)(c[P' \mid a[out\,e.k[out\,e.\overline{h}y]] \mid open\,d] \mid \ldots \\ &\xrightarrow{\tau} (\widetilde{x})(y)(c[P' \mid open\,d] \mid a[k[out\,e.\overline{h}y]] \mid \ldots) \mid e[\overline{out}\,a] \\ &\xrightarrow{\tau} (\widetilde{x})(y)(c[P' \mid open\,d] \mid k[\overline{h}y] \mid a_1[\overline{b}_1x_1] \mid \ldots \mid a_n[\overline{b}_nx_n]) \end{aligned}$$

The reduction must take the following form

$$C[Q] \Longrightarrow (\widetilde{x})(y)(c[Q' \mid open\,d] \mid k[\overline{h}y] \mid a_1[\overline{b}_1x_1] \mid \ldots \mid a_n[\overline{b}_nx_n])$$

According to Corollary 8.4 the local name z is distinct from any of $x_1, \ldots, x_n$.
- $\lambda = b[\overline{a}x_i]$. Let the context $C[\_]$ be defined by

$$(\widetilde{x})(C'[\_] \mid a_1[\overline{b}_1x_1] \mid \ldots \mid a_n[\overline{b}_nx_n]) \mid e[\overline{out}\,c \mid \overline{out}\,a]$$

where $C'[\_]$ is $c[\_ \mid a[b(z).out\,e.k[out\,e.\overline{h}z]] \mid open\,d]$ and e, h, k are fresh. Then the argument is similar to the one in the previous case.

We conclude that $\{\mathcal{R}^{\widetilde{x}}\}_{\widetilde{x}\subseteq_f\mathcal{N}}$ is an open bisimulation. □

Theorem 8.6 is significant in that it justifies Definition 8.2 by supporting the following result.

Theorem 8.7 *In the functional FA the equivalence relations $\approx^f_{opn}$, $\approx^f_{obs}$, $\approx^f_{brb}$ and $\approx^f_{wob}$ are the same.*

Proof The inclusion $\approx^f_{opn} \subseteq \approx^f_{obs}$ follows from Proposition 8.3 and the inclusion $\approx^f_{obs} \subseteq \approx^f_{opn}$ follows from Theorem 8.6. The proofs of $\approx^f_{obs} = \approx^f_{brb} = \approx^f_{wob}$ are almost the same as the ones given in Sect. 7. The idea is to show that $\approx^f_{wob}$ is contained in $\approx^f_{obs}$. The extra work is to do with the functional capabilities. Suppose for example that $Q \approx^f_{wob} P \xrightarrow{\overline{a}(x)} P'$. Let $\widetilde{x}$ be the free names in $P \mid Q$ and d, e be fresh names. Let $C[\_]$ be $(\widetilde{x})(a[\_] \mid a[a(z).\overline{open}\, e.\overline{d}z] \mid e[open\, a.e] \mid e[\overline{e}])$. By the definition of $\approx^f_{wob}$, $C[P] \Longrightarrow (\widetilde{x})(x)(a[P'] \mid e[\overline{d}x])$ has to be simulated by $C[Q] \Longrightarrow (\widetilde{x})(x)(a[Q'] \mid e[\overline{d}x])$ for some Q' or by $C[Q] \Longrightarrow (\widetilde{x})(a[Q'] \mid e[\overline{d}y])$ for some Q' and $y \in \widetilde{x}$. Using similar arguments as in the proofs of Corollaries 8.4 and 8.5, one can rule out the possibility of the reductions $C[Q] \Longrightarrow (\widetilde{x})(a[Q'] \mid e[\overline{d}y])$ as a simulation. Thus $Q \Downarrow \overline{a}(x)$. □

We will write $\approx^f$ for any of $\approx^f_{opn}$, $\approx^f_{obs}$, $\approx^f_{brb}$ and $\approx^f_{wob}$ and we will simply read "$P \approx^f Q$" as "P is bisimilar to Q".

Corollary 8.8 (Local observability) *For functional FA processes P, Q it holds that $P \approx^f Q$ if and only if $C[P] \approx^f C[Q]$ for every ambient context $C[\_]$.*

9 Pi as a subcalculus of FA

The λ-calculus [2] is an operational model of the functional programming. It has been used as a test bed for the descriptive power of a computing model. Milner has given in [27] a translation that relates the λ-calculus and the π-calculus. More precisely Milner's translation interprets the β-reduction of the lazy λ-calculus [1] by the communication of the π-calculus. At the semantic level, the force of the translation is witnessed by the full abstraction result proved by Sangiorgi [36], which states that two λ-terms are open applicative bisimilar if and only if their translations are bisimilar as π-processes.

We prove in this section that the lazy λ-calculus can also be embedded in the functional FA. We shall establish this fact by showing that the π-calculus is a sub-calculus of the functional FA. The syntax and the semantics of the π-calculus is formulated in Fig. 3. This variant of the π-calculus has all the machinery to code up the lazy λ-calculus.

We need to define a map $[\![\_]\!]$ that translates a π-process to an FA system. The crux of the matter is how to interpret the π-processes with the input/output prefixes. Using the terminologies of the functional FA, one can read the π-process $a(x).P$ as 'an ambient that receives a name *from* ambient a before continuing as P', and the π-process $\overline{a}y.Q$ as 'an ambient that sends the name y *to* ambient a before continuing as Q'. This suggests to define $[\![a(x).P]\!]$ by $a[a(x).[\![P]\!]]$ and $[\![\overline{a}y.Q]\!]$ by $a[\overline{a}y.[\![Q]\!]]$. Unfortunately this straightforward translation does not work. The communication would have been modeled by the following reduction:

$$a[a(x).[\![P]\!]] \mid a[\overline{a}y.[\![Q]\!]] \xrightarrow{\tau} a[[\![P\{y/x\}]\!]] \mid a[[\![Q]\!]]$$

The π-Process

$M := \mathbf{0} \mid a(x).P \mid \overline{a}x.P \mid P|P' \mid (x)P \mid !a(x).P$

The Operational Semantics

$$\frac{}{a(x).P \xrightarrow{ay} P\{y/x\}} \qquad \frac{}{\overline{a}x.P \xrightarrow{\overline{a}x} P} \qquad \frac{P \xrightarrow{\overline{a}x} P'}{(x)P \xrightarrow{\overline{a}(x)} P'} \qquad \frac{P \xrightarrow{\lambda} P'}{(x)P \xrightarrow{\lambda} (x)P'}\, x \notin n(\lambda)$$

$$\frac{P \xrightarrow{\lambda} P'}{P|Q \xrightarrow{\lambda} P'|Q} \qquad \frac{P \xrightarrow{ax} P' \quad Q \xrightarrow{\overline{a}x} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \qquad \frac{P \xrightarrow{ax} P' \quad Q \xrightarrow{\overline{a}(x)} Q'}{P|Q \xrightarrow{\tau} (x)(P'|Q')}$$

$$\frac{}{!a(x).P \xrightarrow{ay} P\{y/x\} \mid !a(x).P}$$

Fig. 3 The π-calculus

It is clear that the system on the right hand side is wrong since the two translations stay inside a. Ideally the right hand side should be $[\![P\{y/x\}]\!] \mid [\![Q]\!]$. The unsuccessful attempt leads us to define the translation in such a way that things can be pulled out after a communication has been simulated. We first define an auxiliary translation relative to a fresh name u:

$$\begin{aligned}
[\![\mathbf{0}]\!]_u &\stackrel{\text{def}}{=} \mathbf{0}\\
[\![a(x).P]\!]_u &\stackrel{\text{def}}{=} (v)(a[out\,u.v.a(x).[\![P]\!]_v] \mid v[out\,u.\overline{a}.!\overline{out}\,a])\\
[\![\overline{a}y.Q]\!]_u &\stackrel{\text{def}}{=} (v)(a[out\,u.v.\overline{a}y.[\![Q]\!]_v] \mid v[out\,u.\overline{a}.!\overline{out}\,a])\\
[\![P \mid Q]\!]_u &\stackrel{\text{def}}{=} [\![P]\!]_u \mid [\![Q]\!]_u\\
[\![(x)P]\!]_u &\stackrel{\text{def}}{=} (x)[\![P]\!]_u\\
[\![!a(x).P]\!]_u &\stackrel{\text{def}}{=} (v)(a[out\,u.v.!a(x).[\![P]\!]_v] \mid v[out\,u.\overline{a}.!\overline{out}\,a])
\end{aligned}$$

Intuitively the name u in the above translation is a handle that can be used to pull out the translations as it were from the surrounding ambients. In the definition of $[\![a(x).P]\!]_u$ the two ambients $a[out\,u.v.a(x).[\![P]\!]_v]$ and $v[out\,u.\overline{a}.!\overline{out}\,a]$ must both be pulled out and synchronize before $a[out\,u.v.a(x).[\![P]\!]_v]$ can interact with anybody. The translation would be correct, and even more concise, without the synchronization. But the current translation is easy to reason about. Using the auxiliary definition, we can formally define the translation as follows:

$$\begin{aligned}
[\![\mathbf{0}]\!] &\stackrel{\text{def}}{=} \mathbf{0}\\
[\![a(x).P]\!] &\stackrel{\text{def}}{=} (v)(a[a(x).[\![P]\!]_v] \mid v[!\overline{out}\,a])\\
[\![\overline{a}y.Q]\!] &\stackrel{\text{def}}{=} (v)(a[\overline{a}y.[\![Q]\!]_v] \mid v[!\overline{out}\,a])\\
[\![P \mid Q]\!] &\stackrel{\text{def}}{=} [\![P]\!] \mid [\![Q]\!]\\
[\![(x)P]\!] &\stackrel{\text{def}}{=} (x)[\![P]\!]\\
[\![!a(x).P]\!] &\stackrel{\text{def}}{=} (v)(a[!a(x).[\![P]\!]_v] \mid v[!\overline{out}\,a])
\end{aligned}$$

The encoding is not only structural, it is also stable in the sense of the following lemma, the proof of which is a simple structural induction.

Lemma 11 $[\![P]\!]\sigma \equiv [\![P\sigma]\!]$ *and* $[\![P]\!]_v\sigma \equiv [\![P\sigma]\!]_v$ *if* $v \notin n(\sigma)$.

The stability is a necessary condition for the translation to behave properly on the input prefix processes.

Let's see how to model communications in the functional FA. The interpretation of $a(x).P \mid \overline{a}y.Q$ simulates the π-communication in the very first step:

$$[\![a(x).P \mid \overline{a}y.Q]\!] \xrightarrow{\tau} (v)(a[[\![P\{y/x\}]\!]_v] \mid v[!\overline{out}\,a]) \mid (w)(a[[\![Q]\!]_w] \mid w[!\overline{out}\,a])$$

If $P\{y/x\}$ can not perform any observable actions, then it is bisimilar to $\mathbf{0}$. In this case $(v)(a[[\![P\{y/x\}]\!]_v] \mid v[!\overline{out}\,a])$ is bisimilar to $\mathbf{0}$. If $P\{y/x\}$ is of the form $b(z).P'$ then

$$(v)(a[[\![P\{y/x\}]\!]_v] \mid v[!\overline{out}\,a]) \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}\sim [\![P\{y/x\}]\!]$$

In other words, $[\![P\{y/x\}]\!]_v$ must come out of a before it can simulate $b(z).P'$. This is the general idea of the translation.

By definition the translation $[\![P]\!]$ of a π-process P, and all its descendants can only perform four kinds of actions, the τ-actions, actions of the form $a[ax]$, actions of the form $a[\overline{a}x]$, and actions of the form $(x)a[\overline{a}x]$. These correspond respectively to the four kinds of the actions of the π-calculus.

The first step to understand the translation is to prove the following lemmas.

Lemma 12 *Suppose the only possible observable actions O can perform are either $[out\, v]$ or functional actions. Then the two processes in the following reduction*

$$(v)(a[b[out\, v.P] \mid O] \mid v[!\overline{out}\,a]) \xrightarrow{\tau} (v)(b[P] \mid a[O] \mid v[!\overline{out}\,a])$$

are bisimilar. Moreover an action of $(v)(a[b[out\, v.P] \mid O] \mid v[!\overline{out}\,a])$ can be simulated by $(v)(b[P] \mid a[O] \mid v[!\overline{out}\,a])$ in no more than one action step.

Proof By assumption it is clear that

$$(v)(a[b[out\, v.P] \mid O] \mid v[!\overline{out}\,a]) \xrightarrow{\tau} (v)(b[P] \mid a[O] \mid v[!\overline{out}\,a])$$

Now there is no interactions between $b[out\, v.P]$ and O. And every action of

$$(v)(b[P] \mid a[O] \mid v[!\overline{out}\,a])$$

can be simulated by the same action of $(v)(a[b[out\, v.P] \mid O] \mid v[!\overline{out}\,a])$. On the other hand, if $(v)(a[b[out\, v.P] \mid O] \mid v[!\overline{out}\,a])$ performs an action caused by $b[out\, v.P]$, then the action can be simulated by $(v)(b[P] \mid a[O] \mid v[!\overline{out}\,a])$ vacuously. If

$$(v)(a[b[out\, v.P] \mid O] \mid v[!\overline{out}\,a])$$

performs an action caused by O then $(v)(b[P] \mid a[O] \mid v[!\overline{out}\,a])$ can do the same action. □

Lemma 13 *Suppose P is a π-process and the only possible observable actions O can perform are either $[out\, v]$ or functional actions. Then some P' exists such that*

$$(v)(a[[\![P]\!]_v \mid O] \mid v[!\overline{out}\,a]) \Longrightarrow P' \sim [\![P]\!] \mid (v)(a[O] \mid v[!\overline{out}\,a])$$

and $(v)(a[[\![P]\!]_v \mid O] \mid v[!\overline{out}\,a])$ is bisimilar to $[\![P]\!] \mid (v)(a[O] \mid v[!\overline{out}\,a])$.

Proof The proof is carried out by structural induction.

- P is $b(x).P'$. Then $[\![P]\!]_v$ is $(w)(b[out\, v.w.b(x).[\![P']\!]_w] \mid w[out\, v.\overline{b}.!\overline{out}\, b])$. It is clear that

$$\begin{aligned} & (v)(a[[\![P]\!]_v \mid O] \mid v[!\overline{out}\, a]) \\ \xrightarrow{\tau}\ & (v)((w)(b[w.b(x).[\![P']\!]_w] \mid a[w[out\, v.\overline{b}.!\overline{out}\, b] \mid O]) \mid v[!\overline{out}\, a]) \\ \xrightarrow{\tau}\ & (v)((w)(b[w.b(x).[\![P']\!]_w] \mid w[\overline{b}.!\overline{out}\, b]) \mid a[O] \mid v[!\overline{out}\, a]) \\ \xrightarrow{\tau}\ & (v)((w)(b[b(x).[\![P']\!]_w] \mid w[!\overline{out}\, b]) \mid a[O] \mid v[!\overline{out}\, a]) \\ \xrightarrow{\tau}\ & [\![P]\!] \mid (v)(a[O] \mid v[!\overline{out}\, a]) \end{aligned}$$

  One gets the equivalence between the first and the third processes by applying Lemma 12 to the first two reductions. Notice that the only action the system $(w)(b[w.b(x).[\![P']\!]_w] \mid w[\overline{b}.!\overline{out}\, b])$ can perform is the tau action. Therefore it has to be bisimilar to $(w)(b[b(x).[\![P']\!]_w] \mid w[!\overline{out}\, b])$.
- P is $\overline{b}y.P'$ or $!b(x).P'$. The arguments are similar.
- P is $(x)P'$ or $P' \mid P''$. In these cases we resort to the induction hypothesis.
- P is $\mathbf{0}$. This case is easy.

This completes the proof. □

Lemma 13 is useful since it relates $[\![P]\!]_v$ to $[\![P]\!]$, both operationally and observationally.

The second step is to prove that the translation preserves the operational semantics.

Proposition 9.1 (Preservation) *The translation* $[\![\_]\!]$ *preserves the operational semantics in the following sense*:

(i) *If* $P \xrightarrow{\tau} P'$ *then* $[\![P]\!] \xrightarrow{\tau}\Longrightarrow\sim [\![P']\!]$;

(ii) *If* $P \xrightarrow{ax} P'$ *then* $[\![P]\!] \xrightarrow{a[ax]}\Longrightarrow\sim [\![P']\!]$;

(iii) *If* $P \xrightarrow{\overline{a}x} P'$ *then* $[\![P]\!] \xrightarrow{a[\overline{a}x]}\Longrightarrow\sim [\![P']\!]$;

(iv) *If* $P \xrightarrow{\overline{a}(x)} P'$ *then* $[\![P]\!] \xrightarrow{(x)a[\overline{a}x]}\Longrightarrow\sim [\![P']\!]$.

Proof The proof is carried out by induction on the derivations of reduction. The base case is when P is of the form $a(x).P'$, or $\overline{a}x.P'$ or $!a(x).P'$. Take for instance the process $!a(x).P'$. By Lemma 13

$$\begin{aligned} [\![!a(x).P']\!] & \equiv && (v)(a[!a(x).[\![P']\!]_v] \mid v[!\overline{out}\, a]) \\ & \xrightarrow{a[ay]} && (v)(a[[\![P'\{y/x\}]\!]_v \mid !a(x).[\![P']\!]_v] \mid v[!\overline{out}\, a]) \\ & \Longrightarrow\sim && [\![P'\{y/x\}]\!] \mid [\![!a(x).P']\!] \\ & \equiv && [\![P'\{y/x\} \mid !a(x).P']\!] \end{aligned}$$

The other cases are also simple. □

As a matter of fact the translation is better than what is stated in Proposition 9.1. It actually reflects the operational semantics.

Proposition 9.2 (Reflection) *The translation* $[\![\_]\!]$ *reflects the operational semantics in the following sense*:

(i) *If* $[\![P]\!] \xrightarrow{\tau} S$ *then* $P \xrightarrow{\tau} P'$ *for some* P' *such that* $S \Longrightarrow\sim [\![P']\!]$ *and* $S \approx^f [\![P']\!]$;

(ii) *If* $[\![P]\!] \xrightarrow{a[ax]} S$ *then* $P \xrightarrow{ax} P'$ *for some* P' *such that* $S \Longrightarrow\sim [\![P']\!]$ *and* $S \approx^f [\![P']\!]$;

(iii) *If* $[\![P]\!] \xrightarrow{a[\bar{a}x]} S$ *then* $P \xrightarrow{\bar{a}x} P'$ *for some* P' *such that* $S \Longrightarrow\sim [\![P']\!]$ *and* $S \approx^f [\![P']\!]$;

(iv) *If* $[\![P]\!] \xrightarrow{(x)a[\bar{a}x]} S$ *then* $P \xrightarrow{\bar{a}(x)} P'$ *for some* P' *such that* $S \Longrightarrow\sim [\![P']\!]$ *and* $S \approx^f [\![P']\!]$.

Proof The proof is carried out by mutual structural induction. Suppose $[\![P]\!] \xrightarrow{\tau} S$. Obviously P can be neither a prefix operation nor a replication. If $P \equiv (x)P'$ then $S \equiv (x)S'$ and $[\![P']\!] \xrightarrow{\tau} S'$. We are done by induction hypothesis on (i). If $P \equiv P' \mid P''$ then there are three cases:

- $[\![P]\!] \xrightarrow{\tau} S$ is caused by a tau action of $[\![P']\!]$. We are done by applying the induction hypothesis on (i).
- $[\![P]\!] \xrightarrow{\tau} S$ is caused by a tau action of $[\![P'']\!]$. We are done by applying the induction hypothesis on (i).
- $[\![P]\!] \xrightarrow{\tau} S$ is caused by an interaction between $[\![P']\!]$ and $[\![P'']\!]$. In this case we can use the induction hypothesis on (ii), (iii) and (iv).

For another example, suppose $[\![P]\!] \xrightarrow{a[ax]} S$. If $P \equiv a(z).P'$ then

$$S \equiv (v)(a[[\![P'\{x/z\}]\!]_v] \mid v[!\overline{out}\, a])$$

and

$$[\![P]\!] \xrightarrow{a[ax]} (v)(a[[\![P'\{x/z\}]\!]_v] \mid v[!\overline{out}\, a]) \Longrightarrow\sim [\![P'\{x/z\}]\!]$$

and $(v)(a[[\![P'\{x/z\}]\!]_v] \mid v[!\overline{out}\, a]) \approx^f [\![P'\{x/z\}]\!]$ by using Lemma 13. If P is in any other form then we can simply resort to the induction hypothesis.

The proofs of the other cases are similar. □

The next corollary is a more general statement of the reflection property that is more useful in the reasoning of the bisimulations.

Corollary 9.3 *The translation* $[\![\_]\!]$ *reflects the operational semantics in the following sense*:

(i) *If* $[\![P]\!] \overset{\tau}{\Longrightarrow} S$ *then* $P \overset{\tau}{\Longrightarrow} P'$ *for some* P' *such that* $S \Longrightarrow\sim [\![P']\!]$ *and* $S \approx^f [\![P']\!]$;

(ii) *If* $[\![P]\!] \overset{a[ax]}{\Longrightarrow} S$ *then* $P \overset{ax}{\Longrightarrow} P'$ *for some* P' *such that* $S \Longrightarrow\sim [\![P']\!]$ *and* $S \approx^f [\![P']\!]$;

(iii) *If* $[\![P]\!] \overset{a[\bar{a}x]}{\Longrightarrow} S$ *then* $P \overset{\bar{a}x}{\Longrightarrow} P'$ *for some* P' *such that* $S \Longrightarrow\sim [\![P']\!]$ *and* $S \approx^f [\![P']\!]$;

(iv) *If* $[\![P]\!] \overset{(x)a[\bar{a}x]}{\Longrightarrow} S$ *then* $P \overset{\bar{a}(x)}{\Longrightarrow} P'$ *for some* P' *such that* $S \Longrightarrow\sim [\![P']\!]$ *and* $S \approx^f [\![P']\!]$.

Proof Suppose $[\![P]\!] \xrightarrow{\tau} S_1 \Longrightarrow S$. By Proposition 9.2 some P' exists such that $P \xrightarrow{\tau} P'$ and $S_1 \approx^f [\![P']\!]$. It follows from Lemma 12 that $S_1 \Longrightarrow S$ can be simulated by $[\![P']\!] \Longrightarrow S' \approx^f S$ for some S' in such a way that the length of $[\![P']\!] \Longrightarrow S'$ is no more than the length of $S_1 \Longrightarrow S$. So by induction hypothesis we must have some P'' such that $P' \Longrightarrow P''$ and $S' \approx^f [\![P'']\!]$. Conclude that $P \overset{\tau}{\Longrightarrow} P''$ and $S \approx^f [\![P'']\!]$. This suffices to give the general picture of the proof. □

The close correspondence between the two operational semantics is only half the story. It is apparent that the operational correspondence should support an algebraic correspondence. Therefore the third step is to establish a full abstraction result. In the following theorem, $\approx_{\pi}^{\widetilde{x}}$ denotes Sangiorgi and Walker's quasi open $\widetilde{x}$-bisimilarity. Appendix C reviews some background materials on this equivalence.

Theorem 9.4 (Full abstraction) *Suppose P, Q are π-processes. Then $P \approx_{\pi}^{\widetilde{x}} Q$ if and only if $[\![P]\!] \approx^{\widetilde{x}} [\![Q]\!]$.*

Proof Let $\{\mathcal{R}^{\widetilde{x}}\}_{\widetilde{x}\subseteq_f \mathcal{N}}$ be the family of relations defined by

$$\mathcal{R}^{\widetilde{x}} \stackrel{\text{def}}{=} \{([\![P]\!], [\![Q]\!]) \mid P \approx_{\pi}^{\widetilde{x}} Q \text{ in } \pi\}$$

Suppose $S \approx^{\widetilde{x}} \mathcal{R}^{\widetilde{x}} \approx^{\widetilde{x}} T$. Then $[\![P]\!], [\![Q]\!]$ exist such that

$$S \approx^{\widetilde{x}} [\![P]\!] \, \mathcal{R}^{\widetilde{x}} \, [\![Q]\!] \approx^{\widetilde{x}} T$$

If $S \xrightarrow{(z)a[\overline{a}z]} S'$ then $[\![P]\!] \overset{(z)a[\overline{a}z]}{\Longrightarrow} L$ and $S' \approx^{\widetilde{x}z} L$ for some L. By Corollary 9.3 some P' exists such that $P \overset{\overline{a}(z)}{\Longrightarrow} P'$, $L \Longrightarrow [\![P']\!]$ and $L \approx [\![P']\!]$. Since $P \approx_{\pi}^{\widetilde{x}} Q$, there must exist some Q' such that $Q \overset{\overline{a}(z)}{\Longrightarrow} Q' \approx_{\pi}^{\widetilde{x}z} P'$. By Proposition 9.1, $[\![Q]\!] \overset{(z)a[\overline{a}z]}{\Longrightarrow} N \sim [\![Q']\!]$ for some N. It then follows that $T \overset{(z)a[\overline{a}z]}{\Longrightarrow} T' \approx^{\widetilde{x}z} N$ for some T'. Thus

$$T \overset{(z)a[\overline{a}z]}{\Longrightarrow} T' \approx^{\widetilde{x}z} \mathcal{R}^{\widetilde{x}z} \approx^{\widetilde{x}z} S'$$

The proofs for the other three cases are similar. Conclude that $\{\approx^{\widetilde{x}} \mathcal{R}^{\widetilde{x}} \approx^{\widetilde{x}}\}_{\widetilde{x}\subseteq \mathcal{N}}$ is an open bisimulation. It is then clear that $\mathcal{R}^{\widetilde{x}} \subseteq \approx^{\widetilde{x}} \mathcal{R}^{\widetilde{x}} \approx^{\widetilde{x}} \subseteq \approx^{\widetilde{x}}$. Thus $[\![P]\!] \approx^{\widetilde{x}} [\![Q]\!]$ follows from $P \approx_{\pi}^{\widetilde{x}} Q$.

Let $\{\mathcal{S}^{\widetilde{x}}\}_{\widetilde{x}\subseteq_f \mathcal{N}}$ be the family of relations defined by

$$\mathcal{S}^{\widetilde{x}} \stackrel{\text{def}}{=} \{(P, Q) \mid [\![P]\!] \approx^{\widetilde{x}} [\![Q]\!] \text{ in functional FA}\}$$

We show that it is a quasi open bisimulation. Suppose that $P \xrightarrow{\overline{a}(z)} P'$ in the π-calculus. By Proposition 9.1, one has $[\![P]\!] \xrightarrow{(z)a[\overline{a}z]} P_1 \Longrightarrow\sim [\![P']\!]$ and $P_1 \approx^f [\![P']\!]$ for some FA process P_1. It follows that some FA process Q_1 exists such that $[\![Q]\!] \overset{(z)a[\overline{a}z]}{\Longrightarrow} Q_1 \approx^{\widetilde{x}z} P_1$. By Corollary 9.3, some π-process Q' exists such that $Q \overset{\overline{a}(z)}{\Longrightarrow} Q'$ and $Q_1 \approx^f [\![Q']\!]$. Now we have $[\![P']\!] \approx^{\widetilde{x}z} [\![Q']\!]$. Hence $Q \overset{\overline{a}(z)}{\Longrightarrow} Q' \, \mathcal{S}^{\widetilde{x}z} \, P'$. Conclude that $\{\mathcal{S}^{\widetilde{x}}\}_{\widetilde{x}\subseteq_f \mathcal{N}}$ is a quasi open bisimulation, which means that $P \approx_{\pi}^{\widetilde{x}} Q$ follows from $[\![P]\!] \approx^{\widetilde{x}} [\![Q]\!]$. □

In summary the π-calculus is a sub-calculus of the functional FA. As far as we know this is the best result so far about the relationship between the π-calculus and the ambient calculi.

The full abstraction result is established with respect to the open bisimilarity. To our understanding there is no reason why it can not be proved for other bisimulation equivalences like for instance the early bisimilarity. The details however remain to be verified.

An interesting point about the translation is that, in addition to the functional capabilities, only the out and co-out capabilities are used. That is to say that from the point of view of modeling the π-calculus the in and open capabilities are not necessary. In our opinion the example shows that the out capability is essential for the ambient calculi. Without the operator all the ambients can do is to suck ambients, which would reduce the observability of the ambients considerably.

The translation also suggests that the operational semantics we have chosen for the out capability is reasonable. The local name u in the definitions of $[\![a(x).P]\!]_u$ and $[\![a(x).P]\!]$ for example is crucial. It makes sure that $[\![P]\!]$ can not be interfered by the environment in an uncontrolled way. It only does what it is supposed to do. Had we used the semantics defined in (14) we would have had

$$[\![a(x).P]\!] \overset{\text{def}}{=} (v)(a[a(x).[\![P]\!]_a] \mid v[!\overline{out}\, a]) \tag{28}$$

The problem with (28) is that the component $v[!\overline{out}\, a]$ can interact with the environments, which has to be wrong. We have not been able to work out a sound translation using (14).

10 Conclusion, related work and further research

The philosophy of FA is that computing with ambients is about transferring computing resources from one place to another. The resources could be anything inside an ambient, like an ambient, a process or an interface. The role of the capability operators is to lay down the rules for the transportation of resources. These rules guarantee the subject reduction property for ambients. The investigation of the paper has come up with results that support our computing mechanism. Three aspects are particularly worth mentioning:

1. The operational semantics is cleanly defined by a pure labeled transition system and is subject to the standard bisimulation studies. The simplicity of the operational semantics is largely due to the design principles we set up for FA. On the other hand, the operational simplicity reinforces the principles we proposed for the ambients. The most crucial design decision of FA is concerned with the out capability. It is this new semantics of the out operation that facilitates the nice proof of the congruence property and the coincidence results.
2. The observational theory is supported by the results characterizing the bisimilarity in terms of the observational bisimilarity. It is also enforced by the Local Observability property. Local Observability is in our opinion a fundamental property of ambients that calls for more attention. We know with hindsight that the design of the variants of the ambient calculus has been driven by the desire to separate the observational semantics from the syntax. The success of such an endeavor is manifested by a Local Observability Theorem. The property essentially states that putting a syntactical entity in an ambient neither increases nor decreases its interactive ability. It is a consequence of the well-definedness of the calculus as well as the well-definedness of the observational equivalences. For FA it would be interesting to investigate the validity of the Local Observability property in the absence of the open capability. It is also interesting to see if the property holds for the other variants of the MA when the open capability is ignored.
3. An expressiveness result is established by proving that the π-calculus is a subcalculus of FA. This is in our opinion a major contribution of the paper. It formally relates two well known frameworks of distributed mobile computing, the π framework and the ambient framework. This has enforced our intuition that the π-calculus and the ambient calculi should have a close relationship. As a model for

distributed mobile computing FA encompasses a broader picture of computing than the π-calculus. The interpretation of π gives immediately an interpretation of the lazy λ-calculus in FA. It is our hope that the Full Abstraction Theorem will invite more efforts to investigate the ambient calculi.

The proofs appeared in this paper are interesting not only in that they make extensive use of the Bisimulation lemma but also in that they substantiate our choice of the semantics. Some of the proofs are facilitated by the particular computing mechanism of FA. The proof of Theorem 6.4 for instance depends crucially on the operational semantics we have chosen for the out capability. The proof of the Local Observability property makes full use of the operational semantics of the open capability. From the point of view of the observational semantics, these proofs have also exploited the properties of the bisimulations in an essential way. It is with the help of the bisimilarities that we have been able to discover, and prove, the sub-model relationship between π and FA. In summary the paper offers a proof technique whose prominent feature is the heavy use of the Bisimulation lemma.

FA represents another step towards a better understanding of the calculi for mobile computing. To cast more light on what have been achieved in this paper and what aspects of FA remain to be investigated, we take a look at some related work and issues:

- The algebraic theory of MA has been studied in the framework of reductional semantics [11,19]. This approach is basically based on a reductional operational semantics, a structural congruence and a notion of successful test. Although intuitively straightforward, it is not comfortable to work with the reductional equivalences since one needs to consider all contexts of some form in order to test the equivalence of two ambients. The bisimulation approach to the semantics of the ambient calculi has been studied by several researchers. The SAP of Merro and Hennessy introduced a labeled transition semantics for a variant of SA in [22,23], where the bisimulation theory of SAP was studied. FA appears more streamlined than SAP. The congruence theorem in [22,23] relies on the fact that the capability operators come with passwords. In [24,25] Merro and Zappa Nardelli study the bisimulation equivalence for MA using a two level semantics. The congruence is proved to be the same as the reduction barbed equivalence. It is worth remarking that in the labeled transition semantics of MA the ambient $a[\mathbf{0}]$ can perform an action! This is a clear indication that the ambients of MA are best seen as abbreviations of some variants of MA with co-actions. Vigliotti and Phillips have looked at the barbed congruence in [43] for Safe Ambients. They proved that various definitions of barbedness induce the same congruence. This result is added with more weight by Theorem 7.5.
- The relationship between the π-Calculus and the ambient calculi is a question that has to be settled [30]. Due to its significance, a lot of attentions have been attracted to it. Results that have been obtained previously vary in shape and strength. A minimal requirement for a translation from the π-Calculus to ambient calculi is that the reductional behavior of the former is simulated by that of the latter. The translation in [11] falls in this category. The simulation of the operational semantics may also be required to preserve types [9,12,20]. A translation meeting the minimal condition neither preserves nor reflects the observational equalities between processes. A stronger but reasonable requirement is that the translation reflects the operational semantics. An important consequence of the

stronger property is the full abstraction result about the translation. None of the previous encodings of the (asynchronous) π-Calculus in the ambient calculi satisfies the stronger requirement. The full abstraction result about our translation of the π-Calculus into FA represents a significant progress in the comparative study of the two models of mobile computing.

Analogously previous interpretations of the lazy λ-Calculus are weak in that they do not relate the equivalence of the λ-terms to the equivalence of the interpretations of the λ-terms. Our translation and Milner's encoding of the lazy λ-Calculus immediately give rise to a translation from the lazy λ-Calculus to FA that preserves the β-conversion. In [36] Sangiorgi discovered that Milner's encoding is actually fully abstract with respect to the open applicative bisimilarity, an extension of the applicative bisimilarity of Abramsky [1] to the open λ-terms. It follows that the encoding of the lazy λ-Calculus in FA is also fully abstract with regards to the open applicative bisimilarity.

– Apart from the calculi we have mentioned in the introduction, other variants of MA have been discussed. The Boxed Ambients [4] of Bugliesi, Castagna and Crafa for instance abandon the open capability in favor of more flexible communications. The communications in Boxed Ambients could either be horizontal (between siblings) or vertical (between parents and children). In FA the vertical communications can be achieved by global communications. To get the idea of the encodings it suffices to give the following example:

$$\begin{aligned} a[\downarrow(x).P \mid c[\uparrow y.C]] &\overset{\mathrm{def}}{=} (k)a[k(x).P \mid k[c(v).out.\overline{a}v] \mid c[\overline{k}y.C]] \\ &\xrightarrow{\tau} (k)a[k(x).P \mid k[out.\overline{a}y] \mid c[C]] \\ &\xrightarrow{\tau} (k)(k[\overline{a}y] \mid a[k(x).P \mid c[C]]) \\ &\xrightarrow{\tau} a[P\{y/x\} \mid c[C]] \end{aligned}$$

where *out* is the derived operator defined in Example 3. This is an example of a child sending a message to its parent. Using similar idea one could express the other three forms of vertical communication as well as the local communications of Cardelli and Gordon.

The Controlled Ambients [42] of Teller, Zimmer and Hirschcoff impose the condition that the transmission of an ambient should be consented by both the sender and the receiver. Using the notations developed in this paper the semantics of the Controlled Ambients can be described as follows:

$$a[in\ b.L \mid A] \mid b[\overline{in}\ a.M \mid B] \mid c[letin\ a.N] \xrightarrow{\tau} b[a[L \mid A] \mid M \mid B] \mid c[N]$$
$$a[c[\overline{out}\ b.M \mid B] \mid letout\ c.L \mid A] \mid b[\overline{out}\ a.N \mid C] \xrightarrow{\tau} a[L \mid A] \mid c[M \mid B] \mid b[N \mid C]$$

These three-party interactions can be achieved in two phases: First the present environment consents for an ambient to leave; and second the target environment consents for the ambient to come. This can be done in FA as follows:

$$a[c.in\ b.L \mid A] \mid b[\overline{in}\ a.M \mid B] \mid c[\overline{a}.N] \xrightarrow{\tau}{}^{*} b[a[L \mid A] \mid M \mid B] \mid c[N]$$
$$a[c[\uparrow.out\ b.M \mid B] \mid \downarrow.L \mid A] \mid b[\overline{out}\ a.N \mid C] \xrightarrow{\tau}{}^{*} a[L \mid A] \mid c[M \mid B] \mid b[N \mid C]$$

where we have used the encodings of the vertical communications in FA explained above. This example and the previous one are meant to illustrate how to achieve in FA the various enhancements of the controlling power of ambients proposed in literature.

– Another way to harness the misbehavior of the ambients is to use types. Type systems of almost all the variants of MA have been proposed. The philosophy of types is best explained by the following slogan:

$$\text{Types} = \text{Interfaces} = \text{Protocols}$$

A type specifies the way a process may interact with the environments, which can also be seen as a protocol between the process and the environments. It is a beneficial exercise to look at the type systems for FA.

– There are a number of papers discussing the expressive power of the ambients. The Turing completeness of the ambient calculi have been established in [44,45, 3,5,21]. Maffeis and Phillips proved in [21] that the ambients with only *in* and *out* capabilities but without restrictions are Turing complete. They also showed that termination is a decidable property for the subcalculus with only the *in* (or *out*) capability. In [32] Phillips and Vigliotti showed that the Pure Ambient Calculus without restriction admits symmetric electoral systems. The Full Abstraction Theorem implies that the subcalculus of FA consisting of the out capability and communications is Turing complete. It would be interesting to study the expressive power of the subcalculi of FA in a systematic way.

A lot of interesting issues about FA await studies. We outline some of them below:

– A fundamental question is how much more expressive FA is over π. In other words, is there an encoding from FA to π that preserves and reflects the interactability. The key problem is how to interpret the nested structures (the structure in depth) of the ambients by the flat structure of the π-calculus. It is likely though that the interactional ability of the ambients can not be fully captured by the π-processes.

– Another important question is how orthogonal the capability operators of FA are and how much redundancy there is. To explain what we mean, let FA^{-in} denote the subcalculus of FA that drops the in and co-in operators. If there is an encoding from FA^{-in} to FA, say the inclusion map, that preserves and reflects the interactability, then the *in* and $\overline{in}$ operators are orthogonal to the other three pairs of operators in the sense that the presence of *in* and $\overline{in}$ does not increase the expressiveness of the other three pairs of operators. If on the other hand there is an encoding from FA to FA^{-in} that preserves and reflects the interactability, then the pair $in, \overline{in}$ are redundant in terms of the interactional behavior. In a larger scale the characterizations of the expressiveness of the subcalculi of FA are also important to better our understanding of the ambients. We are currently looking at these issues.

– At a technical level, the question of the coincidence of the late semantics with the early semantics is nontrivial. Here the analogy with the higher order π-calculus [35,38,41] breaks down. The reason that the early semantics and the late semantics coincide for the higher order π-calculus is that one can always prefix a transmitted higher order variable to control its firing after it lands on the target places. On the other hand there is no way to stop a transmitted ambient from action once it is in position.

– A question hard to answer is concerned with the axiomatization of $\approx$ for the finite FA. Ambient calculi are of higher order. The Turing completeness of the higher order communications, like that in the higher order π-calculus, defeats any attempt to obtain a (decidable) complete system. But if we restricts to linear higher order calculus, complete system can be achieved. In [16] a complete equational system is constructed for the Linear Higher Order π. The ambient interactions of FA are linear in the sense that a transmitting ambient is not duplicated when it arrives at the target ambient. Despite of the linearality though, the problem remains untouched.

– From a programming point of view, efforts are called for to see how FA, with or without syntactical sugars, can be used as a core language for web applications, protocol designs, security related issues. In these applications it is extremely useful to have the *if_then_else* construct. This construct does not really make sense in the ambient calculi previously studied. But thanks to the global communications, FA can be extended with the processes of the form

$$\textbf{if } \varphi \textbf{ then } M \textbf{ else } N$$

where φ is a conjunction of match $x = y$ and mismatch $x \neq y$. In this aspect FA opens up a new avenue for applying the ambients to practical programming.

The reports on the progress of attacking some of the above issues would have to occur elsewhere.

Acknowledgment The author is supported by The National Distinguished Young Scientist Fund of NNSFC (60225012), The National 973 Project (2003CB317005) and The National Nature Science Foundation of China (60473006 and 60573002). He wishes to thank X. Cai, X. Dong, H. Long and H. Zhu for their proof reading. Examples 7, 10 and 13 were pointed to us by X. Dong. Example 8 was pointed to us by H. Lv. The author is grateful to the referees for their constructive comments and suggestions.

Appendix A: Two auxiliary definitions

The "bisimulation up to" is a technique widely used in process theory [26]. This is the approach to establish bisimulation equalities up to some known equivalence. In this section we recall two well-known (bi)simulation-up-to relations used in the paper. We first define the strong bisimilarity.

Definition A.1 A symmetric relation $\mathcal{R}$ on processes is a strong bisimulation if it is closed under substitution and the followings hold whenever $P\mathcal{R}Q$:

1. If $P \xrightarrow{\lambda} P'$ and λ is in the set $\{\tau,\kappa\} \cup \{a[\overline{out}\,b], a[[out\,b]], a[bx], a[\overline{b}x], (x)a[\overline{b}x] \mid a,b,x \in \mathcal{N}\}$ then $Q \xrightarrow{\lambda} Q'\mathcal{R}P'$ for some Q'.
2. If $P \xrightarrow{[out\,a]} (\widetilde{m})\langle A\rangle P'$ then for every name d and every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ there exist some $\widetilde{n}, B, Q'$ such that $Q \xrightarrow{[out\,a]} (\widetilde{n})\langle B\rangle Q'$ and $(\widetilde{n})(B \mid d[Q' \mid O])\ \mathcal{R}\ (\widetilde{m})(A \mid d[P' \mid O])$.
3. If $P \xrightarrow{a[in\,b]} (\widetilde{m})\langle A\rangle P'$ then for every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ there exist some $\widetilde{n}, B, Q'$ such that $Q \xrightarrow{a[in\,b]} (\widetilde{n})\langle B\rangle Q'$ and $(\widetilde{n})(Q' \mid b[B \mid O])\ \mathcal{R}\ (\widetilde{m})(P' \mid b[A \mid O])$.

4. If $P \xrightarrow{a[\overline{open}\,b]} (\widetilde{m})\langle M\rangle P'$ then for every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ there exist some $\widetilde{n}, N, Q'$ such that $Q \xrightarrow{a[\overline{open}\,b]} (\widetilde{n})\langle N\rangle Q'$ and $(\widetilde{n})(Q' \mid b[N \mid O])\ \mathcal{R}\ (\widetilde{m})(P' \mid b[M \mid O])$.
5. If $P \xrightarrow{a[\overline{in}\,b]} C[\_]$ then for every N satisfying $fn(N) \cap bn(C[\_]) = \emptyset$ there exists some $C'[\_]$ such that $Q \xrightarrow{a[\overline{in}\,b]} C'[\_]$ and $C'[b[N]]\ \mathcal{R}\ C[b[N]]$.
6. If $P \xrightarrow{a[open\,b]} C[\_]$ then for every N satisfying $fn(N) \cap bn(C[\_]) = \emptyset$ there exists some $C'[\_]$ such that $Q \xrightarrow{a[open\,b]} C'[\_]$ and $C'[N]\ \mathcal{R}\ C[N]$.

The strong bisimilarity $\sim$ is the largest strong bisimulation.

The basic algebraic properties of $\sim$ are guaranteed by the following proposition.

Proposition A.2 *The strong bisimilarity $\sim$ is a congruence.*

In observational theories the equivalence $\sim$ will be explored to iron out the difference between structurally equivalent processes. Technically this is done by using the proof method of 'bisimulation up to $\sim$'. The definition of the 'bisimulation up to $\sim$' and the proof of the next lemma can be found in [26]. In our framework the definition is as follows:

Definition A.3 A family of symmetric relations $\{\mathcal{R}^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f \mathcal{N}}$ on processes is an open simulation up to $\sim$ if whenever $P\mathcal{R}^{\widetilde{z}}Q$ and σ respects $\widetilde{z}$ then $P\sigma\mathcal{R}^{\widetilde{z}}Q\sigma$ and the followings hold:

1. If $P \xrightarrow{\tau} P'$ then $Q \Longrightarrow Q' \sim \mathcal{R}^{\widetilde{z}} \sim P'$ for some Q'.
2. If $P \xrightarrow{\kappa} P'$ and κ is not of the form $\overline{a}(x)$, then $Q \overset{\kappa}{\Longrightarrow} Q' \sim \mathcal{R}^{\widetilde{z}} \sim P'$ for some Q'.
3. If $P \xrightarrow{\overline{a}(x)} P'$ then $Q \overset{\overline{a}(x)}{\Longrightarrow} Q' \sim \mathcal{R}^{\widetilde{z}x} \sim P'$ for some Q'.
4. If $P \xrightarrow{\lambda} P'$ and λ is a label in the following set $\{a[\overline{out}\,b], a[[out\,b]], a[bx], a[\overline{b}x] \mid a, b \in \mathcal{N}\}$ then $Q \overset{\lambda}{\Longrightarrow} Q' \sim \mathcal{R}^{\widetilde{z}} \sim P'$ for some Q'.
5. If $P \xrightarrow{(x)a[\overline{b}x]} P'$ then $Q \overset{(x)a[\overline{b}x]}{\Longrightarrow} Q' \sim \mathcal{R}^{\widetilde{z}x} \sim P'$ for some Q'.
6. If $P \xrightarrow{[out\,a]} (\widetilde{m})\langle A\rangle P'$ then for every name d and every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ there exist some $\widetilde{n}, B, Q', Q''$ such that $Q \Longrightarrow\xrightarrow{[out\,a]} (\widetilde{n})\langle B\rangle Q'$ and $(\widetilde{n})(B \mid d[Q' \mid O]) \Longrightarrow Q'' \sim \mathcal{R}^{\widetilde{z}} \sim (\widetilde{m})(A \mid d[P' \mid O])$.
7. If $P \xrightarrow{a[in\,b]} (\widetilde{m})\langle A\rangle P'$ then for every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ some $\widetilde{n}, B, Q', Q''$ exist such that $Q \Longrightarrow\xrightarrow{a[in\,b]} (\widetilde{n})\langle B\rangle Q'$ and $(\widetilde{n})(Q' \mid b[B \mid O]) \Longrightarrow Q'' \sim \mathcal{R}^{\widetilde{z}} \sim (\widetilde{m})(P' \mid b[A \mid O])$.
8. If $P \xrightarrow{a[\overline{open}\,b]} (\widetilde{m})\langle M\rangle P'$ then for every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ some $\widetilde{n}, N, Q', Q''$ exist such that $Q \Longrightarrow\xrightarrow{a[\overline{open}\,b]} (\widetilde{n})\langle N\rangle Q'$ and $(\widetilde{n})(Q' \mid b[N \mid O]) \Longrightarrow Q'' \sim \mathcal{R}^{\widetilde{z}} \sim (\widetilde{m})(P' \mid b[M \mid O])$.
9. If $P \xrightarrow{a[\overline{in}\,b]} C[\_]$ then for every N satisfying $fn(N) \cap bn(C[\_]) = \emptyset$ there exist some $C'[\_], Q'$ such that $Q \Longrightarrow\xrightarrow{a[\overline{in}\,b]} C'[\_]$ and $C'[b[N]] \Longrightarrow Q' \sim \mathcal{R}^{\widetilde{z}} \sim C[b[N]]$.
10. If $P \xrightarrow{a[open\,b]} C[\_]$ then for every N satisfying $fn(N) \cap bn(C[\_]) = \emptyset$ there exist some $C'[\_], Q'$ such that $Q \Longrightarrow\xrightarrow{a[open\,b]} C'[\_]$ and $C'[N] \Longrightarrow Q' \sim \mathcal{R}^{\widetilde{z}} \sim C[N]$.

Proposition A.4 *If $\{\mathcal{R}^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ is an open bisimulation up to $\sim$ then $\mathcal{R}^{\widetilde{z}} \subseteq \approx^{\widetilde{z}}$ for all finite sequences $\widetilde{z}$ of names.*

As is pointed out in [39] (open) bisimulations up to $\approx$ fail a proposition like Proposition A.4. Many useful alternatives have been proposed though. The next definition provides one such alternative.

Definition A.5 A family of symmetric relations $\{\mathcal{R}^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ on processes is an open simulation up to $\{\approx^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ if whenever $P\mathcal{R}^{\widetilde{z}}Q$ and σ respects $\widetilde{z}$ then $P\sigma\mathcal{R}^{\widetilde{z}}Q\sigma$ and the followings hold:

1. If $P \xrightarrow{\tau} P'$ then $Q \Longrightarrow Q' \approx^{\widetilde{z}} \mathcal{R}^{\widetilde{z}}P'$ for some Q'.
2. If $P \xrightarrow{\kappa} P'$ and κ is not of the form $\overline{a}(x)$, then $Q \overset{\kappa}{\Longrightarrow} Q' \approx^{\widetilde{z}} \mathcal{R}^{\widetilde{z}}P'$ for some Q'.
3. If $P \xrightarrow{\overline{a}(x)} P'$ then $Q \overset{\overline{a}(x)}{\Longrightarrow} Q' \approx^{\widetilde{z}x} \mathcal{R}^{\widetilde{z}x}P'$ for some Q'.
4. If $P \xrightarrow{\lambda} P'$ and λ is a label in the following set $\{a[\overline{out}\, b], a[[out\, b]], a[bx], a[\overline{b}x] \mid a, b \in \mathcal{N}\}$ then $Q \overset{\lambda}{\Longrightarrow} Q' \approx^{\widetilde{z}} \mathcal{R}^{\widetilde{z}}P'$ for some Q'.
5. If $P \xrightarrow{(x)a[\overline{b}x]} P'$ then $Q \overset{(x)a[\overline{b}x]}{\Longrightarrow} Q' \approx^{\widetilde{z}x} \mathcal{R}^{\widetilde{z}x}P'$ for some Q'.
6. If $P \xrightarrow{[out\, a]} (\widetilde{m})\langle A\rangle P'$ then for every name d and every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ there exist some $\widetilde{n}, B, Q', Q''$ such that $Q \Longrightarrow\xrightarrow{[out\, a]} (\widetilde{n})\langle B\rangle Q'$ and $(\widetilde{n})(B \mid d[Q' \mid O]) \Longrightarrow Q'' \approx^{\widetilde{z}} \mathcal{R}^{\widetilde{z}} (\widetilde{m})(A \mid d[P' \mid O])$.
7. If $P \xrightarrow{a[in\, b]} (\widetilde{m})\langle A\rangle P'$ then for every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ some $\widetilde{n}, B, Q', Q''$ exist such that $Q \Longrightarrow\xrightarrow{a[in\, b]} (\widetilde{n})\langle B\rangle Q'$ and $(\widetilde{n})(Q' \mid b[B \mid O]) \Longrightarrow Q'' \approx^{\widetilde{z}} \mathcal{R}^{\widetilde{z}} (\widetilde{m})(P' \mid b[A \mid O])$.
8. If $P \xrightarrow{a[\overline{open}\, b]} (\widetilde{m})\langle M\rangle P'$ then for every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ some $\widetilde{n}, N, Q', Q''$ exist such that $Q \Longrightarrow\xrightarrow{a[\overline{open}\, b]} (\widetilde{n})\langle N\rangle Q'$ and $(\widetilde{n})(Q' \mid b[N \mid O]) \Longrightarrow Q'' \approx^{\widetilde{z}} \mathcal{R}^{\widetilde{z}} (\widetilde{m})(P' \mid b[M \mid O])$.
9. If $P \xrightarrow{a[\overline{in}\, b]} C[\_]$ then for every N satisfying $fn(N) \cap bn(C[\_]) = \emptyset$ there exist some $C'[\_]$, Q' such that $Q \Longrightarrow\xrightarrow{a[\overline{in}\, b]} C'[\_]$ and $C'[b[N]] \Longrightarrow Q' \approx^{\widetilde{z}} \mathcal{R}^{\widetilde{z}} C[b[N]]$.
10. If $P \xrightarrow{a[open\, b]} C[\_]$ then for every N satisfying $fn(N) \cap bn(C[\_]) = \emptyset$ there exist some $C'[\_]$, Q' such that $Q \Longrightarrow\xrightarrow{a[open\, b]} C'[\_]$ and $C'[N] \Longrightarrow Q' \approx^{\widetilde{z}} \mathcal{R}^{\widetilde{z}} C[N]$.

By using the following lemma, the above definition can be exploited to establish bisimulation equalities.

Proposition A.6 *If both $\{\mathcal{R}^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ and $\{(\mathcal{R}^{\widetilde{z}})^{-1}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ are open simulations up to $\{\approx^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ then $\mathcal{R}^{\widetilde{z}} \subseteq \approx^{\widetilde{z}}$ for all finite sequences $\widetilde{z}$ of names.*

Proof Suppose for instance $P \approx^{\widetilde{z}} \mathcal{R}^{\widetilde{z}} \approx^{\widetilde{z}} Q \xrightarrow{a[in\, b]} (\tilde{x})\langle A\rangle N$. By repeated use of the definitions of $\mathcal{R}^{\widetilde{z}}$ and $\approx^{\widetilde{z}}$, it is not difficult to see that for every O some $\tilde{y}, B, M, P'$ exist such that $P \Longrightarrow\xrightarrow{a[in\, b]} (\tilde{y})\langle B\rangle M$ and

$$(\widetilde{y})(M \mid b[B \mid O]) \Longrightarrow P' \approx^{\widetilde{z}} \mathcal{R}^{\widetilde{z}} \approx^{\widetilde{z}} (\widetilde{m})(N \mid b[A \mid O])$$

In this manner one could prove that $\{\approx^{\widetilde{z}} \mathcal{R}^{\widetilde{z}} \approx^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ is a bisimulation. □

Appendix B: The proof of Proposition 8.3

In this appendix we prove that $\approx^{f}_{opn}$ is a congruence equivalence. The idea of the proof is the same as the one in [22]. Before proving the result, we define an order relation on transitions.

Suppose $t = (t_1, \ldots, t_n)$ is a tree with subtrees $t_1, \ldots, t_n$ and s is another tree. We say that s is order less than t if either of the following properties hold:

- the depth of s is less than the depth of t;
- $s = (t_1, \ldots, t_{i-1}, s_i, t_{i+1}, \ldots, t_n)$ and s_i is order less than t_i.

The tree order is the transitive closure of the order less relation. Now the derivations can be ordered by imposing the tree order on them. Since every transition has a unique derivation, we thus have an order relation on the transitions.

The following lemma states the congruence property of the $\widetilde{x}$-bisimilarity.

Lemma 14 *The following properties hold:*

(i) *If $P \approx^{\widetilde{x}} Q$ then $\kappa.P \approx^{\widetilde{x}} \kappa.Q$ provided that κ is not of the form $a(x)$;*

(ii) *If $P \approx^{\widetilde{x}} Q$ then $P \,|\, O \approx^{\widetilde{x}} Q \,|\, O$;*

(iii) *If $P \approx^{\widetilde{x}} Q$ then $a[P] \approx^{\widetilde{x}} a[Q]$;*

(iv) *If $P \approx^{\widetilde{x}} Q$ then $(y)P \approx^{\widetilde{x}} (y)Q$;*

(v) *If $P \approx^{\widetilde{x}} Q$ then $!P \approx^{\widetilde{x}} !Q$.*

Proof (i) The proof is straightforward.

(ii,iii,iv) Construct a series of relations as follows:

$$\mathcal{S}_0^{\widetilde{z}} \stackrel{\text{def}}{=} \approx^{f}_{opn}$$

$$\vdots$$

$$\mathcal{S}_{i+1}^{\widetilde{z}} \stackrel{\text{def}}{=} \left\{ \begin{array}{c} (P|N, Q|N) \\ ((x)P, (x)Q) \\ (a[P], a[Q]) \end{array} \middle| \, P\, \mathcal{S}_i^{\widetilde{z}}\, Q \right\}$$

$$\vdots$$

Let $\mathcal{S}^{\widetilde{z}}$ be $\bigcup_{i\in\omega} \mathcal{S}_i^{\widetilde{z}}$. Clearly $\mathcal{S}^{\widetilde{z}}$ is the relation

$$\{(C[P], C[Q]) \mid P \approx^{\widetilde{z}} Q,\ C[\_] \text{ a context}\}$$

We prove by structural induction that $\{\mathcal{S}^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f \mathcal{N}}$ is an open bisimulation up to $\sim$. The inductive proof is carried out in two steps:

1. *Base Step*: The pairs in $\mathcal{S}_0^{\widetilde{z}}$ satisfy the bisimulation up to $\sim$ property;
2. *Induction Step*: If the pairs in $\mathcal{S}_i^{\widetilde{z}}$ satisfy the bisimulation up to $\sim$ property, then the pairs in $\mathcal{S}_{i+1}^{\widetilde{z}}$ satisfy the bisimulation up to $\sim$ property.

Since $\approx^{f}_{opn}$ satisfies the bisimulation property by definition, we only have to prove the induction step. Notice that we are proving that $\mathcal{S}^{\widetilde{z}}$ is an open bisimulation up to $\sim$ by stratifying the elements of $\mathcal{S}^{\widetilde{z}}$.

Now suppose that the pairs in $\mathcal{S}_i^{\tilde{z}}$ satisfy the bisimulation up to $\sim$ property and that $P\mathcal{S}_i^{\tilde{z}}Q$. That is we have the following *induction hypotheses*:

1. If $P \xrightarrow{\tau} P'$ then $Q \Longrightarrow Q' \sim \mathcal{S}^{\tilde{z}} \sim P'$ for some Q'.
2. If $P \xrightarrow{\kappa} P'$ and κ is not of the form $\overline{a}(x)$, then $Q \overset{\kappa}{\Longrightarrow} Q' \sim \mathcal{S}^{\tilde{z}} \sim P'$ for some Q'.
3. If $P \xrightarrow{\overline{a}(x)} P'$ then $Q \overset{\overline{a}(x)}{\Longrightarrow} Q' \sim \mathcal{S}^{\tilde{z}x} \sim P'$ for some Q'.
4. If $P \xrightarrow{\lambda} P'$ and λ is a label in the following set $\{a[\overline{out}\, b], a[[out\, b]], a[bx], a[\overline{b}x] \mid a, b \in \mathcal{N}\}$ then $Q \overset{\lambda}{\Longrightarrow} Q' \sim \mathcal{S}^{\tilde{z}} \sim P'$ for some Q'.
5. If $P \xrightarrow{(x)a[\overline{b}x]} P'$ then $Q \overset{(x)a[\overline{b}x]}{\Longrightarrow} Q' \sim \mathcal{S}^{\tilde{z}x} \sim P'$ for some Q'.
6. If $P \xrightarrow{[out\, a]} (\widetilde{m})\langle A\rangle P'$ then for every name d and every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ there exist some $\widetilde{n}, B, Q', Q''$ such that $Q \Longrightarrow\xrightarrow{[out\, a]} (\widetilde{n})\langle B\rangle Q'$ and $(\widetilde{n})(B \mid d[Q' \mid O]) \Longrightarrow Q'' \sim \mathcal{S}^{\tilde{z}} \sim (\widetilde{m})(A \mid d[P' \mid O])$.
7. If $P \xrightarrow{a[in\, b]} (\widetilde{m})\langle A\rangle P'$ then for every process O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ some $\widetilde{n}, B, Q', Q''$ exist such that $Q \Longrightarrow\xrightarrow{a[in\, b]} (\widetilde{n})\langle B\rangle Q'$ and $(\widetilde{n})(Q' \mid b[B \mid O]) \Longrightarrow Q'' \sim \mathcal{S}^{\tilde{z}} \sim (\widetilde{m})(P' \mid b[A \mid O])$.
8. If $P \xrightarrow{a[\overline{open}\, b]} (\widetilde{m})\langle M\rangle P'$ then for every O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ some $\widetilde{n}, N, Q'$, Q'' exist such that $Q \Longrightarrow\xrightarrow{a[\overline{open}\, b]} (\widetilde{n})\langle N\rangle Q'$ and $(\widetilde{n})(Q' \mid b[N \mid O]) \Longrightarrow Q'' \sim \mathcal{S}^{\tilde{z}} \sim (\widetilde{m})(P' \mid b[M \mid O])$.
9. If $P \xrightarrow{a[\overline{in}\, b]} C[\_]$ then for every N satisfying $fn(N) \cap bn(C[\_]) = \emptyset$ there exist some $C'[\_]$, Q' such that $Q \Longrightarrow\xrightarrow{a[\overline{in}\, b]} C'[\_]$ and $C'[b[N]] \Longrightarrow Q' \sim \mathcal{S}^{\tilde{z}} \sim C[b[N]]$.
10. If $P \xrightarrow{a[open\, b]} C[\_]$ then for every N satisfying $fn(N) \cap bn(C[\_]) = \emptyset$ there exist some $C'[\_]$, Q' such that $Q \Longrightarrow\xrightarrow{a[open\, b]} C'[\_]$ and $C'[N] \Longrightarrow Q' \sim \mathcal{S}^{\tilde{z}} \sim C[N]$.

We need to verify that the pairs $(P|N, Q|N)$, $((x)P, (x)Q)$, $(a[P], a[Q])$, which are in $\mathcal{S}_{i+1}^{\tilde{z}}$, satisfy the bisimulation up to $\sim$ property. In other words, we demonstrate that each of the above ten bisimulation properties persists through structural composition. In what follows, we enumerate for each structural composition, say $(P|N, Q|N)$, all the possible interactive effects of each of the above ten bisimulations and show that the interactions always result in bisimulations under the induction hypotheses.

– $(P|L, Q|L)$.

  1. $P \xrightarrow{\tau} P'$. $P \mid L \xrightarrow{\tau} P' \mid L$ is simulated by $Q \mid L \Longrightarrow Q' \mid L \sim \mathcal{S}^{\tilde{z}} \sim P' \mid L$.
  2. $P \xrightarrow{\kappa} P'$ and κ is not of the form $\overline{a}(x)$. $P \mid L \xrightarrow{\kappa} P' \mid L$ is simulated by $Q \mid L \overset{\kappa}{\Longrightarrow} Q' \mid L \sim \mathcal{S}^{\tilde{z}} \sim P' \mid L$.
  3. $P \xrightarrow{\overline{a}(x)} P'$. $P|L \xrightarrow{\overline{a}(x)} P'|L$ is simulated by $Q|L \overset{\overline{a}(x)}{\Longrightarrow} Q'|L \sim \mathcal{S}^{\tilde{z}x} \sim P'|L$.
  4. $P \xrightarrow{\lambda} P'$ and λ is a label in the following set $\{a[\overline{out}\, b], a[[out\, b]], a[bx], a[\overline{b}x] \mid a, b \in \mathcal{N}\}$. $P \mid L \xrightarrow{\lambda} P' \mid L$ is simulated by $Q \mid L \overset{\lambda}{\Longrightarrow} Q' \mid L \sim \mathcal{S}^{\tilde{z}} \sim P' \mid L$.
  5. $P \xrightarrow{(x)a[\overline{b}x]} P'$. $P|L \xrightarrow{(x)a[\overline{b}x]} P'|L$ is simulated by $Q|L \overset{(x)a[\overline{b}x]}{\Longrightarrow} Q'|L \sim \mathcal{S}^{\tilde{z}x} \sim P'|L$.
  6. $P \xrightarrow{[out\, a]} (\widetilde{m})\langle A\rangle P'$. $P \mid L \xrightarrow{[out\, a]} (\widetilde{m})\langle A\rangle(P' \mid L)$ is simulated by $Q \mid L \Longrightarrow\xrightarrow{[out\, a]} (\widetilde{n})\langle B\rangle(Q' \mid L)$ and $(\widetilde{n})(B \mid d[Q' \mid L \mid O]) \Longrightarrow Q''$ such that $Q'' \sim \mathcal{S}^{\tilde{z}} \sim (\widetilde{m})(A \mid d[P' \mid L \mid O])$.

7. $P \xrightarrow{a[in\, b]} (\widetilde{m})\langle A\rangle P'$. Two cases:

   - $P \mid L \xrightarrow{a[in\, b]} (\widetilde{m})\langle A\rangle(P' \mid L)$ is simulated by $Q \mid L \Longrightarrow\xrightarrow{a[in\, b]} (\widetilde{n})\langle B\rangle(Q' \mid L)$ and

$$\begin{aligned}(\widetilde{n})(Q' \mid L \mid b[B \mid O]) &\sim (\widetilde{n})(Q' \mid b[B \mid O]) \mid L \\ &\Longrightarrow Q'' \mid L \\ &\mathcal{S}^{\tilde{z}}\ (\widetilde{m})(P' \mid b[A \mid O]) \mid L \\ &\sim (\widetilde{m})(P' \mid L \mid b[A \mid O])\end{aligned}$$

   - $L \xrightarrow{b[\overline{in}\, a]} C[\_]$. In this case $P \mid L \xrightarrow{\tau} (\widetilde{m})(P' \mid C[A])$ is simulated by

$$\begin{aligned}Q \mid L &\overset{\tau}{\Longrightarrow} (\widetilde{n})(Q' \mid C[B]) \\ &\Longrightarrow\sim Q'' \\ &\sim \mathcal{S}^{\tilde{z}} \sim (\widetilde{m})(P' \mid C[A])\end{aligned}$$

8. $P \xrightarrow{a[\overline{open}\, b]} (\widetilde{m})\langle M\rangle P'$. Two cases:

   - $P|L \xrightarrow{a[\overline{open}\, b]} (\widetilde{m})\langle M\rangle(P'|L)$ is simulated by $Q|L \Longrightarrow\xrightarrow{a[\overline{open}\, b]} (\widetilde{n})\langle N\rangle(Q'|L)$ and

$$\begin{aligned}(\widetilde{n})(Q' \mid L \mid b[N \mid O]) &\sim (\widetilde{n})(Q' \mid b[N \mid O]) \mid L \\ &\Longrightarrow Q'' \mid L \\ &\mathcal{S}^{\tilde{z}}\ (\widetilde{m})(P' \mid b[M \mid O]) \mid L \\ &\sim (\widetilde{m})(P' \mid L \mid b[M \mid O])\end{aligned}$$

   - $L \xrightarrow{b[open\, a]} C[\_]$. In this case $P \mid L \xrightarrow{\tau} (\widetilde{m})(P' \mid C[M])$ is simulated by

$$\begin{aligned}Q \mid L &\overset{\tau}{\Longrightarrow} (\widetilde{n})(Q' \mid C[N]) \\ &\Longrightarrow\sim Q'' \\ &\sim \mathcal{S}^{\tilde{z}} \sim (\widetilde{m})(P' \mid C[M])\end{aligned}$$

9. $P \xrightarrow{a[\overline{in}\, b]} C[\_]$. Two cases:

   - $P \mid L \xrightarrow{a[\overline{in}\, b]} C[\_] \mid L$ is simulated by $Q \mid L \Longrightarrow\xrightarrow{a[\overline{in}\, b]} C'[\_] \mid L$ and

$$C'[b[N]] \mid L \Longrightarrow Q' \mid L \sim \mathcal{S}^{\tilde{z}} \sim C[b[N]] \mid L$$

   - $L \xrightarrow{b[in\, a]} (\widetilde{x})\langle B\rangle L'$. $P \mid L \xrightarrow{\tau} (\widetilde{x})(C[B] \mid L')$ is simulated by

$$\begin{aligned}Q \mid L &\Longrightarrow (\widetilde{x})(C'[B] \mid L') \\ &\Longrightarrow (\widetilde{x})(Q' \mid L') \\ &\sim \mathcal{S}^{\tilde{z}} \sim (\widetilde{x})(C[B] \mid L')\end{aligned}$$

10. $P \xrightarrow{a[open\, b]} C[\_]$. Two cases:

   - $P \mid L \xrightarrow{a[open\, b]} C[\_] \mid L$ is simulated by $Q \mid L \Longrightarrow\xrightarrow{a[open\, b]} C'[\_] \mid L$ and

$$C'[N] \mid L \Longrightarrow Q' \mid L \sim \mathcal{S}^{\tilde{z}} \sim C[N] \mid L$$

– $L \overset{b[\overline{open}\,a]}{\longrightarrow} (\widetilde{x})\langle M\rangle L'$. $P \mid L \overset{\tau}{\longrightarrow} (\widetilde{x})(C[M] \mid L')$ is simulated by

$$\begin{array}{rcl} Q \mid L & \Longrightarrow & (\widetilde{x})(C'[N] \mid L') \\ & \Longrightarrow & (\widetilde{x})(Q' \mid L') \\ & \sim \mathcal{S}^{\widetilde{z}} \sim & (\widetilde{x})(C[M] \mid L') \end{array}$$

– $((v)P, (v)Q)$.

1. $P \overset{\tau}{\longrightarrow} P'$. $(v)P \overset{\tau}{\longrightarrow} (v)P'$ is simulated by $(v)Q \Longrightarrow (v)Q' \sim \mathcal{S}^{\widetilde{z}} \sim (v)P'$.
2. $P \overset{\kappa}{\longrightarrow} P'$ and κ is not of the form $\overline{a}(x)$. $(x)P \overset{\kappa}{\longrightarrow} (x)P'$ is simulated by $(x)Q \overset{\kappa}{\Longrightarrow} (x)Q' \sim (x)P'$.
3. $P \overset{\overline{a}(x)}{\longrightarrow} P'$. $(v)P \overset{\overline{a}(x)}{\longrightarrow} (v)P'$ is simulated by $(v)Q \overset{\overline{a}(x)}{\Longrightarrow} (v)Q' \sim \mathcal{S}^{\widetilde{z}x} \sim (v)P'$.
4. $P \overset{\lambda}{\longrightarrow} P'$ and λ is a label in the following set $\{a[\overline{out}\,b], a[[out\,b]], a[bx], a[\overline{b}x] \mid a, b \in \mathcal{N}\}$. And $(v)P \overset{\lambda}{\longrightarrow} (v)P'$ is simulated by $(v)Q \overset{\lambda}{\Longrightarrow} (v)Q' \sim \mathcal{S}^{\widetilde{z}} \sim (v)P'$.
5. $P \overset{(x)a[\overline{b}x]}{\longrightarrow} P'$. $(v)P \overset{(x)a[\overline{b}x]}{\longrightarrow} (v)P'$ is simulated by $(v)Q \overset{(x)a[\overline{b}x]}{\Longrightarrow} (v)Q' \sim \mathcal{S}^{\widetilde{z}x} \sim (v)P'$.
6. $P \overset{[out\,a]}{\longrightarrow} (\widetilde{m})\langle A\rangle P'$. $(v)P \overset{[out\,a]}{\longrightarrow} (v)(\widetilde{m})\langle A\rangle P'$ or $(v)P \overset{[out\,a]}{\longrightarrow} (\widetilde{m})\langle A\rangle(v)P'$ is simulated either by $(v)Q \Longrightarrow\overset{[out\,a]}{\longrightarrow} (v)(\widetilde{n})\langle B\rangle Q'$ and

$$(v)(\widetilde{n})(B \mid d[Q' \mid O]) \Longrightarrow (v)Q'' \sim \mathcal{S}^{\widetilde{z}} \sim (v)(\widetilde{m})(A \mid d[P' \mid O])$$

or by $(v)Q \Longrightarrow\overset{[out\,a]}{\longrightarrow} (\widetilde{n})\langle B\rangle(v)Q'$ and

$$\begin{array}{rcl} (\widetilde{n})(B \mid d[(v)Q' \mid O]) & \sim & (v)(\widetilde{n})(B \mid d[Q' \mid O]) \\ & \Longrightarrow & (v)Q'' \\ & \sim \mathcal{S}^{\widetilde{z}} \sim & (v)(\widetilde{m})(A \mid d[P' \mid O]) \end{array}$$

7. $P \overset{a[in\,b]}{\longrightarrow} (\widetilde{m})\langle A\rangle P'$. $(v)P \overset{a[in\,b]}{\longrightarrow} (v)(\widetilde{m})\langle A\rangle P'$ or $(v)P \overset{a[in\,b]}{\longrightarrow} (\widetilde{m})\langle A\rangle(v)P'$ is simulated either by $(v)Q \Longrightarrow\overset{a[in\,b]}{\longrightarrow} (v)(\widetilde{n})\langle B\rangle Q'$ and

$$(v)(\widetilde{n})(Q' \mid b[B \mid O]) \Longrightarrow (v)Q'' \sim \mathcal{S}^{\widetilde{z}} \sim (v)(\widetilde{m})(P' \mid b[A \mid O])$$

or by $(v)Q \Longrightarrow\overset{a[in\,b]}{\longrightarrow} (\widetilde{n})\langle B\rangle(v)Q'$ and

$$\begin{array}{rcl} (\widetilde{n})((v)Q' \mid b[B \mid O]) & \sim & (v)(\widetilde{n})(Q' \mid b[B \mid O]) \\ & \Longrightarrow & (v)Q'' \\ & \sim \mathcal{S}^{\widetilde{z}} \sim & (v)(\widetilde{m})(P' \mid b[A \mid O]) \end{array}$$

8. $P \overset{a[\overline{open}\,b]}{\longrightarrow} (\widetilde{m})\langle M\rangle P'$. $(v)P \overset{a[\overline{open}\,b]}{\longrightarrow} (v)(\widetilde{m})\langle M\rangle P'$ or $(v)P \overset{a[in\,b]}{\longrightarrow} (\widetilde{m})\langle M\rangle(v)P'$ is simulated either by $(v)Q \Longrightarrow\overset{a[\overline{open}\,b]}{\longrightarrow} (v)(\widetilde{n})\langle N\rangle Q'$ and

$$(v)(\widetilde{n})(Q' \mid b[N \mid O]) \Longrightarrow (v)Q'' \sim \mathcal{S}^{\widetilde{z}} \sim (v)(\widetilde{m})(P' \mid b[M \mid O])$$

or by $(v)Q \Longrightarrow\overset{a[\overline{open}\,b]}{\longrightarrow} (\widetilde{n})\langle N\rangle(v)Q'$ and

$$\begin{array}{rcl} (\widetilde{n})((v)Q' \mid b[N \mid O]) & \sim & (v)(\widetilde{n})(Q' \mid b[N \mid O]) \\ & \Longrightarrow & (v)Q'' \\ & \sim \mathcal{S}^{\widetilde{z}} \sim & (v)(\widetilde{m})(P' \mid b[M \mid O]) \end{array}$$

9. $P \xrightarrow{a[\overline{in}\,b]} C[\_]$. $(\nu)P \xrightarrow{a[\overline{in}\,b]} (\nu)C[\_]$ is simulated by $(\nu)Q \Longrightarrow\xrightarrow{a[\overline{in}\,b]} (\nu)C'[\_]$ and
$$(\nu)C[b[B]] \Longrightarrow (\nu)Q' \sim \mathcal{S}^{\tilde{z}} \sim (\nu)C[b[B]]$$
10. $P \xrightarrow{a[\overline{in}\,b]} C[\_]$. $(\nu)P \xrightarrow{a[\overline{in}\,b]} (\nu)C[\_]$ is simulated by $(\nu)Q \Longrightarrow\xrightarrow{a[\overline{in}\,b]} (\nu)C'[\_]$ and
$$(\nu)C[N] \Longrightarrow (\nu)Q' \sim \mathcal{S}^{\tilde{z}} \sim (\nu)C[N]$$

– $(c[P], c[Q])$.

1. $P \xrightarrow{\tau} P'$. $c[P] \xrightarrow{\tau} c[P']$ is simulated by $c[Q] \Longrightarrow c[Q'] \sim \mathcal{S}^{\tilde{z}} \sim c[P']$.
2. $P \xrightarrow{\kappa} P'$. Six cases:
   - $P \xrightarrow{in\,a} P'$. $c[P] \xrightarrow{c[in\,a]} \langle P'\rangle\mathbf{0}$ is simulated by $c[Q] \Longrightarrow\xrightarrow{c[in\,a]} \langle Q_1\rangle\mathbf{0}$ and
   $$a[c[Q_1] \mid O] \Longrightarrow a[c[Q'] \mid O] \sim \mathcal{S}^{\tilde{z}} \sim a[c[P'] \mid O]$$
   - $P \xrightarrow{\overline{open}\,a} P'$. $c[P] \xrightarrow{c[\overline{open}\,a]} \langle P'\rangle\mathbf{0}$ is simulated by $c[Q] \Longrightarrow\xrightarrow{c[\overline{open}\,a]} \langle Q_1\rangle\mathbf{0}$ and
   $$a[Q_1 \mid O] \Longrightarrow a[Q' \mid O] \sim \mathcal{S}^{\tilde{z}} \sim a[P' \mid O]$$
   - $P \xrightarrow{\overline{in}\,a} P'$. $c[P] \xrightarrow{c[\overline{in}\,a]} c[\_ \mid P']$ is simulated by $c[Q] \Longrightarrow\xrightarrow{c[\overline{in}\,a]} c[\_ \mid Q_1]$ and
   $$c[a[A] \mid Q_1] \Longrightarrow c[a[A] \mid Q'] \sim \mathcal{S}^{\tilde{z}} \sim c[a[A] \mid P']$$
   - $P \xrightarrow{open\,a} P'$. $c[P] \xrightarrow{c[open\,a]} c[\_ \mid P']$ is simulated by $c[Q] \Longrightarrow\xrightarrow{c[open\,a]} c[\_ \mid Q_1]$ and
   $$c[M \mid Q_1] \Longrightarrow c[M \mid Q'] \sim \mathcal{S}^{\tilde{z}} \sim c[M \mid P']$$
   - $P \xrightarrow{out\,a} P'$. $c[P] \xrightarrow{c[out\,a]} \langle P'\rangle\mathbf{0}$ is simulated by $c[Q] \Longrightarrow\xrightarrow{c[out\,a]} \langle Q_1\rangle\mathbf{0}$ and
   $$c[Q_1] \mid a[O] \Longrightarrow c[Q'] \mid a[O] \sim \mathcal{S}^{\tilde{z}} \sim c[P'] \mid a[O]$$
   since $c[Q'] \sim \mathcal{S}^{\tilde{z}} \sim c[P']$.
   - $P \xrightarrow{\overline{out}\,a} P'$. $c[P] \xrightarrow{c[\overline{out}\,a]} c[P']$ is simulated by $c[Q] \overset{c[\overline{out}\,a]}{\Longrightarrow} c[Q'] \sim \mathcal{S}^{\tilde{z}} \sim c[P']$.
   - $P \xrightarrow{ax} P'$. $c[P] \xrightarrow{c[ax]} c[P']$ is simulated by $c[Q] \overset{c[ax]}{\Longrightarrow} c[Q'] \sim \mathcal{S}^{\tilde{z}} \sim c[P']$.
   - $P \xrightarrow{\overline{a}x} P'$. $c[P] \xrightarrow{c[\overline{a}x]} c[P']$ is simulated by $c[Q] \overset{c[\overline{a}x]}{\Longrightarrow} c[Q'] \sim \mathcal{S}^{\tilde{z}} \sim c[P']$.
3. $P \xrightarrow{\overline{a}(x)} P'$. $b[P] \xrightarrow{(x)b[\overline{a}x]} b[P']$ is simulated by $b[Q] \overset{(x)b[\overline{a}x]}{\Longrightarrow} b[Q'] \sim \mathcal{S}^{\tilde{z}x} \sim b[P']$.
4. $P \xrightarrow{\lambda} P'$ and λ is a label in the following set $\{a[\overline{out}\,b], a[[out\,b]], a[bx], a[\overline{b}x] \mid a, b \in \mathcal{N}\}$. This action is prohibited by $c[P]$.
5. $P \xrightarrow{(x)a[\overline{b}x]} P'$. This is prohibited by $c[P]$.
6. $P \xrightarrow{[out\,b]} (\widetilde{m})\langle A\rangle P'$. In this case the action $c[P] \xrightarrow{c[[out\,b]]} (\widetilde{m})(A \mid c[P'])$ is simulated by
$$\begin{aligned} c[Q] &\overset{c[[out\,b]]}{\Longrightarrow} (\widetilde{n})(B \mid c[Q']) \\ &\Longrightarrow Q'' \\ &\sim \mathcal{S}^{\tilde{z}} \sim (\widetilde{m})(A \mid c[P']) \end{aligned}$$
7. $P \xrightarrow{a[in\,b]} (\widetilde{m})\langle A\rangle P'$. This is prohibited by $c[P]$.

8. $P \overset{a[\overline{open}\,b]}{\longrightarrow} (\widetilde{m})\langle M\rangle P'$. This is prohibited by $c[P]$.
9. $P \overset{a[\overline{in}\,b]}{\longrightarrow} C[\_]$. This is prohibited by $c[P]$.
10. $P \overset{a[open\,b]}{\longrightarrow} C[\_]$. This is prohibited by $c[P]$.

We have proved that all pairs in $\mathcal{S}^{\widetilde{z}}$ satisfy the bisimulation up to $\sim$ property defined in Definition A.1. Thus $\mathcal{S}^{\widetilde{z}} \subseteq \approx^{\widetilde{z}}$ by Proposition A.4.

(v) Let $\mathcal{R}^{\widetilde{x}}$ be the following binary relation

$$\{(C[!P], C[!Q]) \mid P \approx^{\widetilde{x}} Q,\ C[\_] \text{ a context}\}$$

We prove that $\{\mathcal{R}^{\widetilde{x}}\}_{\widetilde{z}\subseteq_f \mathcal{N}}$is an open simulation up to $\{\approx^{\widetilde{x}}\}_{\widetilde{z}\subseteq_f \mathcal{N}}$. This is done by using induction on the derivations. Now suppose $C[!P]\ \mathcal{R}^{\widetilde{x}}\ C[!Q]$ and

$$C[!P] \overset{a[in\,b]}{\longrightarrow} (\widetilde{m})\langle A\rangle P' \tag{29}$$

If the action (29) is solely caused by the context, then the bisimulation is easy to establish. If P participates in the above action then

$$C[P \mid !P] \overset{a[in\,b]}{\longrightarrow} (\widetilde{m})\langle A\rangle P' \tag{30}$$

with a smaller derivation order. By induction hypothesis it follows that for each O some $\widetilde{n}, B, Q', Q''$ exist such that the action in (30) can be matched up by

$$C[P \mid !Q] \Longrightarrow\overset{a[in\,b]}{\longrightarrow} (\widetilde{n})\langle B\rangle Q' \tag{31}$$

and

$$(\widetilde{n})(Q' \mid b[B \mid O]) \Longrightarrow Q''\ \mathcal{R}^{\widetilde{x}}\ (\widetilde{m})(P' \mid b[A \mid O])$$

But $C[P \mid !Q] \approx^{\widetilde{x}} C[Q \mid !Q] \sim C[!Q]$ by (ii) through (iv). Therefore (31) can be simulated by $C[!Q]$. That is to say that some $\widetilde{n_1}, B_1, Q_1', Q_1''$ exist such that $C[!Q] \Longrightarrow\overset{a[in\,b]}{\longrightarrow} (\widetilde{n_1})\langle B_1\rangle Q_1'$ and

$$(\widetilde{n_1})(Q_1' \mid b[B_1 \mid O]) \Longrightarrow Q_1''\ \approx^{\widetilde{x}}\ \mathcal{R}^{\widetilde{x}}\ (\widetilde{m})(P' \mid b[A \mid O])$$

The arguments for the other actions are similar.

The intuition behind the above proof is this: Since every action of $!P$ is caused by a finite number of P's, the action in (29) is essentially the action $C[P \mid \cdots \mid P \mid !P] \overset{\lambda}{\longrightarrow} U_1$ in which $!P$ does not participate in the action. Then the following observation

$$C[!P] \sim C[P \mid \cdots \mid P \mid !P]\ \mathcal{R}^{\widetilde{x}}\ C[P \mid \cdots \mid P \mid !Q] \approx^{\widetilde{x}} C[!Q]$$

immediately suggests the proof.

So we have established that both $\{\mathcal{S}^{\widetilde{x}}\}_{\widetilde{z}\subseteq_f \mathcal{N}}$ and $\{(\mathcal{S}^{\widetilde{x}})^{-1}\}_{\widetilde{z}\subseteq_f \mathcal{N}}$ are open simulations up to $\{\approx^{\widetilde{x}}\}_{\widetilde{z}\subseteq_f \mathcal{N}}$. We are done by applying Proposition A.6. $\square$

Proposition 8.3 then follows from the above lemma and the following two lemmas.

Lemma 15 *If $P \approx^{\widetilde{x}} Q$ then $P \approx^{\widetilde{y}} Q$ whenever $\widetilde{x} \subseteq \widetilde{y}$.*

Proof Let $\{\mathcal{R}^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f \mathcal{N}}$ be the family of relations where each $\mathcal{R}^{\widetilde{z}}$ is defined by the following set:

$$\{(P,Q) | P \approx^{\widetilde{y}} Q \text{ whenever } \widetilde{y} \subseteq \widetilde{z}\}$$

It is routine to show that $\{\mathcal{R}^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f \mathcal{N}}$ is an open bisimulation. □

Lemma 16 *If $P \approx^{\widetilde{x}y} Q$ for $y \notin fn(P|Q)$ then $P \approx^{\widetilde{x}} Q$.*

Proof Let $\{\mathcal{R}^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f \mathcal{N}}$ be the family of relations where each $\mathcal{R}^{\widetilde{z}}$ is defined by the following set:

$$\{(P\sigma, Q\sigma) | P \approx^{\widetilde{z}y} Q,\ y \notin fn(P|Q) \text{ and } \sigma \text{ respects } \widetilde{z}\}$$

Now suppose $P \approx^{\widetilde{z}y} Q$, $y \notin fn(P|Q)$ and σ respects $\widetilde{z}$. There are three cases:

- σ does not contain y. Then σ respects $\widetilde{z}y$. This case is simple because $P\sigma \approx^{\widetilde{z}y} Q\sigma$.
- y is in the domain of σ but not in the range of σ. In this case $P\sigma \equiv P\sigma'$ and $Q\sigma \equiv Q\sigma'$ for some σ' that respects $\widetilde{z}y$. So this case is also simple.
- y is in the range of σ. In this case there are some substitution σ' and some name x such that $\sigma = \sigma'\{y/x\}$ and σ' respects $\widetilde{z}y$. So the problem is reduced to cases where substitutions are of the form $\{y/x\}$. Now suppose $\sigma = \{y/x\}$ and $P\sigma \xrightarrow{a\sigma[in\, b\sigma]} (\widetilde{m})\langle A\sigma\rangle P'\sigma$. Since $y \notin fn(P)$, it is clear that $P \xrightarrow{a[in\, b]} (\widetilde{m})\langle A\rangle P'$. Then for O satisfying $\widetilde{m} \cap fn(O) = \emptyset$ there exist $\widetilde{n}, B, Q', Q''$ such that $Q \Longrightarrow\xrightarrow{a[in\, b]} (\widetilde{n})\langle B\rangle Q'$ and

$$(\widetilde{n})(Q' \mid b[B \mid O]) \Longrightarrow Q'' \approx^{\widetilde{z}y} (\widetilde{m})(P' \mid b[A \mid O])$$

  Thus $Q\sigma \Longrightarrow\xrightarrow{a\sigma[in\, b\sigma]} (\widetilde{n\sigma})\langle B\sigma\rangle Q'\sigma$ and

$$(\widetilde{n})(Q' \mid b[B \mid O])\sigma \Longrightarrow Q''\sigma\ \mathcal{R}^{\widetilde{z}}\ (\widetilde{m})(P' \mid b[A \mid O])\sigma$$

  Other cases can be proved similarly.

We are done. □

Proposition B.1 *The equivalence $\approx^{f}_{opn}$ is congruent.*

Proof Since $\approx^{f}_{opn}$ is closed under respectful substitutions, it is closed under all substitutions. Therefore it is closed under the prefix $a(x)$ operation. The rest follows from Lemma 14. □

Appendix C: Quasi open bisimulation

An interesting example concerning the open bisimilarity is given by Sangiorgi and Walker [40]. Consider the pair of the π-processes:

$$C \stackrel{\text{def}}{=} (z)\overline{a}z.(a(w) + a(w).\overline{z}z + a(w).[w{=}z]\overline{z}z)$$
$$D \stackrel{\text{def}}{=} (z)\overline{a}z.(a(w) + a(w).\overline{z}z)$$

According to the definition of the open bisimulation [37] the transitions $C \xrightarrow{\overline{a}(z)}\xrightarrow{a(w)} [w{=}z]\overline{z}z$ can not be simulated by D. But from an observational viewpoint, C and D should be equivalent.

The quasi open bisimulations are rectifications of the open bisimulations that remove the inequalities like the one in the above. Hence they are weaker than the open bisimulations. Here is the definition.

Definition C.1 A quasi open bisimulation is a family of symmetric relations $\{\mathcal{R}^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ on the π-processes such that, for all P and Q, if $P\mathcal{R}^{\widetilde{z}}Q$ and σ respects $\widetilde{z}$ then $P\sigma\mathcal{R}^{\widetilde{z}}Q\sigma$ and the following properties hold:

(i) If $P \xrightarrow{\tau} P'$ then $Q \Longrightarrow Q'\mathcal{R}^{\widetilde{z}}P'$ for some Q'.
(ii) If $P \xrightarrow{ax} P'$ then $Q \overset{ax}{\Longrightarrow} Q'\mathcal{R}^{\widetilde{z}}P'$ for some Q'.
(iii) If $P \xrightarrow{\overline{a}x} P'$ then $Q \overset{\overline{a}x}{\Longrightarrow} Q'\mathcal{R}^{\widetilde{z}}P'$ for some Q'.
(iv) If $P \xrightarrow{\overline{a}(x)} P'$ then $Q \overset{\overline{a}(x)}{\Longrightarrow} Q'\mathcal{R}^{\widetilde{z}x}P'$ for some Q'.

We write $\{\approx_\pi^{\widetilde{z}}\}_{\widetilde{z}\subseteq_f\mathcal{N}}$ for the largest quasi open bisimulation and refer to $\approx_\pi^{\widetilde{z}}$ as the quasi open $\widetilde{z}$-bisimilarity. The quasi open bisimilarity $\approx_\pi$ is the quasi open $\emptyset$-bisimilarity $\approx_\pi^{\emptyset}$.

The quasi open bisimilarity $\approx_\pi$ is the same as the open barbed bisimilarity, the latter being defined as the largest barbed bisimulation closed under substitution. The closure property of the quasi open bisimilarity is guaranteed by the following lemma.

Lemma 17 *The following properties hold:*

(i) *If $P \approx_\pi^{\widetilde{x}} Q$ then $\overline{a}y.P \approx_\pi^{\widetilde{x}} \overline{a}y.Q$ and $\tau.P \approx_\pi^{\widetilde{x}} \tau.Q$;*
(ii) *If $P \approx_\pi^{\widetilde{x}} Q$ then $P \mid O \approx_\pi^{\widetilde{x}} Q \mid O$;*
(iii) *If $P \approx_\pi^{\widetilde{x}} Q$ and $y \notin \widetilde{x}$ then $(y)P \approx_\pi^{\widetilde{x}} (y)Q$;*
(iv) *If $P \approx_\pi^{\widetilde{x}y} Q$ then $(y)P \approx_\pi^{\widetilde{x}} (y)Q$;*
(v) *If $P \approx_\pi^{\widetilde{x}} Q$ then $!a(x).P \approx_\pi^{\widetilde{x}} !a(x).Q$.*

The congruence of $\approx_\pi$ follows from the above lemma and the fact that $\approx_\pi$ is closed under respectful substitutions and therefore all substitutions.

Corollary C.2 *$\approx_\pi$ is equivalent and congruent.*

For more information on the quasi open bisimilarity, see [40,15].

References

1. Abramsky, S.: The lazy lambda calculus. In: Turner, D. (ed.) Declarative Programming, pp. 65–116. Addison-Wesley (1988)
2. Barendregt, H.: The lambda calculus: its syntax and semantics, Studies in Logic and Foundations of Mathematics, North-Holland (1984)
3. Boneva, I., Talbot, J.: When ambients cannot be opened. In: Gordon, A. (ed.) FoSSaCS 2003, pp. 169–184, Lecture Notes in Computer Science, vol. 2620 (2003)
4. Bugliesi, M., Castagna, G., Crafa, S.: Boxed ambients. In: Kobayashi, N., Pierce, B. (eds.) TACS 2001, pp. 38–63, Lecture Notes in Computer Science, vol. 2215 (2001)
5. Busi, N., Zavattaro, G.: On the expressive power of movement and restriction in pure mobile ambients. Theor. Comput. Sci. **322**, 477–515 (2004)
6. Cardelli, L.: Abstraction for mobile computation. In: Secure Internet Programming: Security Issues for Mobile and Distributed Object, pp. 51–94, Lecture Notes in Computer Science, vol. 1603 (1999)
7. Cardelli, L.: Brane calculi. In: Danos, V., Schächter, V. (eds.) International Conference on Computational Methods in System Biology, Paris, May 26–28, 2004, Lecture Notes in Computer Science, vol. 3082 (2005)

8. Cardelli, L., Gordon, A.: Mobile ambients. In: Nivat, M. (ed.) FoSSaCS 1998, pp. 140–155 Lecture Notes in Computer Science, vol. 1378, (1998)
9. Cardelli, L., Gordon, A.: Types for mobile ambients. In: POPL 1999, pp. 79–92. ACM (1999)
10. Cardelli, L., Gordon, A.: Anytime, anywhere: modal logics for mobile ambients. In: POPL 2000, pp. 365–377. ACM (2000)
11. Cardelli, L., Gordon, A.: Mobile ambients. Theor. Comput. Sci. **240**, 177–213 (2000)
12. Cardelli, L., Ghelli, G., Gordon, A.: Mobility types for mobile ambients. In: Wiedermann, J., Emde Boas, P., Nielsen, M. (eds.) ICALP 1999, pp. 230–239, Lecture Notes in Computer Science (1999)
13. Fu, Y.: Variations on mobile processes. Theor. Comput. Sci. **221**, 327–368 (1999)
14. Fu, Y.: Bisimulation congruences of chi calculus. Inf. Comput. **184**, 201–226 (2003)
15. Fu, Y.: On quasi open bisimulation. Theor. Comput. Sci. **338**, 96–126 (2005)
16. Fu, Y.: Checking equivalences for higher order process, working paper (2006)
17. Fu, Y., Yang, Z.: Tau laws for pi calculus. Theor. Comput. Sci. **308**, 55–130 (2003)
18. Guan, X., Yang, Y., You, J.: Typing evolving ambients. Inf. Process. Lett. **80**, 265–270 (2001)
19. Gordon, A., Cardelli, L.: Equational properties of mobile ambients. Math. Struct. Comput. Sci. **13**(3), 371–408 (2003)
20. Levi, F., Sangiorgi, D.: Controlling interference in ambients. In: POPL 2000, pp. 352–364. ACM (2000)
21. Maffeis, S., Phillips, I.: On the computational strength of pure ambient calculi. Theor. Comput. Sci. **330**, 501–551 (2005)
22. Merro, M., Hennessy, M.: Bisimulation congruences in safe ambients. In: POPL 2002, pp. 71–80. ACM (2002)
23. Merro, M., Hennessy, M.: A bisimulation-based semantic theory of safe ambients. ACM Trans. Program. Languages Systems **28**(2), 290–330 (2006)
24. Merro, M., Zappa Nardelli, F.: Bisimulation proof methods for mobile ambients. In: Baeten, J., Lenstra, J., Parrow, J., Woeginger, G. (eds.) ICALP 2003, pp. 584–598, Lecture Notes in Computer Science, vol. 2719 (2003)
25. Merro, M., Zappa Nardelli, F.: Behavioural theory for mobile ambients. J. ACM **52**(6), 961–1023 (2005)
26. Milner, R.: Communication and Concurrency. Prentice Hall, New Jersey (1989)
27. Milner, R.: Functions as processes. Math. Struct. Comput. Sci. **2**, 119–146 (1992)
28. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes. In: Information and Computation, vol. 100, pp. 1–40 (Part I), pp. 41–77 (Part II). Academic, New York (1992)
29. Milner, R., Sangiorgi, D.: Barbed bisimulation. In: Dershowitz, N., Lindenstrauss, N. (eds.) ICALP 1992, pp. 685–695, Lecture Notes in Computer Science, vol. 623 (1992)
30. Nestmann, U.: Welcome to the jungle: a subjective guide to mobile process calculi. In: Bauer, C., Hermanns, H. (eds.) CONCUR 2006, pp. 52–63, Lecture Notes in Computer Science, vol. 4137 (2006)
31. Phillips, I., Vigliotti, M.: On reduction semantics for the push and pull ambient calculus. In: TCS 2002, IFIP 17th World Computer Congress, Montreal. Kluwer (2002)
32. Phillips, I., Vigliotti, M.: Electoral systems in ambient calculi. In: Walukiewicz, I. (ed.) FoSSaCS 2004, pp. 408–422, Lecture Notes in Computer Science, vol. 2987 (2004)
33. Regev, A., Panina, E., Silverman, W., Cardelli, L., Shapiro, E.: Bioambients: an abstraction for biological compartments. Theor. Comput. Sci. **325**(1), 141–167 (2004)
34. Sangiorgi, D.: Expressing mobility in process algebras: first-order and higher-order paradigms, PhD thesis, University of Edinburgh (1993)
35. Sangiorgi, D.: From π-calculus to higher order π-calculus-and back. In: Gaudel, M., Jouannaud, J. (eds.) TAPSOFT'93, pp. 151-166, Lecture Notes in Computer Science, vol. 668 (1993)
36. Sangiorgi, D.: The lazy lambda calculus in a concurrency scenario. Inf. Comput. **111**, 120–153 (1994)
37. Sangiorgi, D.: A theory of bisimulation for π-calculus. Acta Inf. **3**, 69–97 (1996)
38. Sangiorgi, D.: Bisimulation for higher order process calculi. Inf. Comput. **131**(2), 141–178 (1996)
39. Sangiorgi, D., Milner, R.: Techniques of "Weak Bisimulation Up To". In: Cleaveland, W. (ed.) CONCUR 1992, pp. 32–46, Lecture Notes in Computer Science, vol. 630 (1992)
40. Sangiorgi, D., Walker, D.: On barbed equivalence in π-calculus. In: Larsen, K., Nielsen, M. (eds.) CONCUR 2001, pp. 292–304, Lecture Notes in Computer Science, vol. 2154 (2001)
41. Sangiorgi, D., Walker, D.: The Pi Calculus—A Theory of Mobile Processes. CUP (2001)

42. Teller, D., Zimmer, P., Hirschcoff, D.: Using ambients to control resources. In: Brim, L., Jancar, P., Kretinsky, M., Kucera, A. (eds.) CONCUR 2002, pp. 288–303, Lecture Notes in Computer Science, vol. 2421 (2002)
43. Vigliotti, M., Phillips, I.: Barbs and congruemces for safe mobile ambients. Electr. Notes Theor. Comput. Sci. 66(3) (2002)
44. Zimmer, P.: On the expressiveness of pure mobile ambients. Electr. Notes Theor. Comput. Sci. 39(1) (2000)
45. Zimmer, P.: On the expressiveness of pure safe ambients. Math. Struct. Comput. Sci. **13**(5), 721–770 (2003)