Imperial College of Science,
Technology and Medicine
(University of London)

Department of Computing

Reduction Semantics for Ambient Calculi

Maria Grazia Vigliotti

A thesis submitted for the degree of
Doctor in Philosophy of the University of London
and for the Diploma of Imperial College

January 2004

Preface

First of all, I would like to thank my supervisor Dr. Iain C. C. Phillips, for his support and collaboration during this period of research. I thank Iain for having taught me to be more precise and sharp, and for long, detailed and inspiring discussions on the topic of this dissertation. Finally I thank him for his enormous patience towards my stubbornness.

I would like to thank Dr. Nobuko Yoshida for many useful discussions and for being very supportive and positive about my work.

To Sergio Maffeis go thanks for many discussions on various subjects of research and philosophy during the last two years at Imperial College. He suggested an improvement to the solution for the leader election problem for the Ambient Calculus.

I would like to thank also Andrew Phillips, and the concurrency group at Imperial for the Monday lunch meetings. This has been a wonderful forum for discussing various aspects of my work.

I like to thank Prof. Chris Hankin and Dr. Sophia Drossopoulou for helping me on various occasions with administrative problems and (especially Chris) for supporting most of my travelling.

I do not know how I could have ever achieved this without my husband, Steffen van Bakel. He has always believed in me and never failed to support and encourage me, and he has endured my presence in the last months of this Ph.D. Moreover he has read part of this thesis and provided useful comments. His help with LaTeX has been invaluable. He drew the diagrams in this thesis, and got me out trouble with LaTeX many times.

The E.P.S.R.C. has provided me with a Ph.D. grant which I hope I have honoured by producing high quality research.

Sarah Cooke from Imperial helped me to deal with the ‘ghosts’ related to the submission of this thesis.

This thesis is dedicated to a very intelligent woman, my mother Speranza Savastano in Vigliotti, who was borne at a time and location of great regression, when women were not allowed to study. She impressed on me, at a very young age, the importance of education. I will never forget the evenings, after a long day of work, she spent helping me with my homework during my primary education.

****

This thesis has been greatly improved by the corrections suggested by my two examiners: Dr. A.D. Gordon and Dr. K. Honda. I deeply thank them both for the care taken in evaluating my work.

To my mother, Speranza Savastano

Contents

Notation

Syntax of processes

$\mathbf{0}$	Inactive process	18
$P \mid Q$	Composition	18
$n\,[P]$	Ambient process	26
$(\nu n)P$	Creation of fresh names	26
$!P$	Replication	18
$[n=m]P$	Matching	18
$a(n).P$	Input on channel a	18
$\overline{a}\langle n\rangle$	Output on channel a	18
$\sum_{i\in I}\alpha_i.P$	Mixed choice	18
$\sum_{i\in I}\alpha_i^I.P$	Input choice	24
$\sum_{i\in I}\alpha_i^O.P$	Output choice	24
$(n).P$	Anonymous input	26
$\langle n\rangle$	Anonymous output	26
$M.P$	Prefix	26
$\mathsf{in}\,n$	In capability	26
$\mathsf{out}\,n$	Out capability	26
$\mathsf{open}\,n$	Open capability	26
$\mathsf{pull}\,n$	Pull capability	60
$\mathsf{push}\,n$	Push capability	60
$\overline{\mathsf{in}}\,n$	In co-capability	34
$\overline{\mathsf{open}}\,n$	Open co-capability	34
$\overline{\mathsf{out}}\,n$	Out co-capability	34
$\mathit{fn}(P)$	Free names in P	19
$\mathit{bn}(P)$	Bound names in P	19

Semantics

Sound Theories

Calculi

Introduction

Models for concurrency, communication and/or distributed systems aim to describe, in an abstract and manageable way, computation that involves processes computing in parallel (concurrency), interacting processes (communication), and processes computing at different places (distribution). A real-life example of a distributed system is the network of the Department of Computing of Imperial College. A network of this kind is called a *local area network*. For concurrent and communicating systems, well established models have been proposed in the last few decades, like CSP [44], CCS [54] and ACP [7]. These calculi have no primitives that describe locations, although such behaviour could be simulated with the right semantics [11]. Lots of research has been carried out to understand distributed algorithms; for a good overview we refer the reader to [50, 80, 4].

The Internet behaves as a distributed system, in the sense specified above; however, it poses new challenges and new problems to computer science, as argued by people working in networking [29] and by theoreticians [18, 19]. This means that traditional models or algorithms devised for standard distributed systems might not work for the Internet.

On the practical side, there exists the problem of managing the Internet. As an example, Estrin considers Inter-Administrative Routing protocols [29]. That article, written in the early days of the Internet, already identifies problems in standard routing algorithms. The Internet is described there as "*interconnection of technically heterogeneous local area networks, (...) interconnection of autonomous Administrative Domains*". An administrative domain is defined as a "*region that is governed by a single authority*". Local area networks and administrative domains may not coincide. For instance, IBM can be seen as one administrative domain that possesses several local area networks spread across the earth. Different administrative domains enforce different policies that affect traditional design of routing protocols. Estrin argues that previous routing protocols would either work on a single administrative domain or support a too small range of policies, thus the need for new routing protocols that take heterogeneous administrative domain into account.

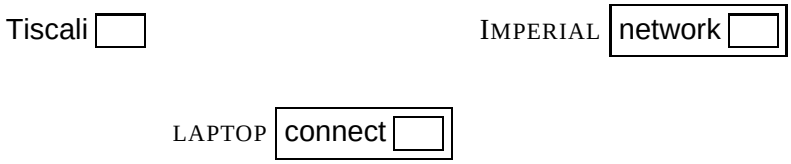
On the theoretical side of computer science, Cardelli [18, 19] argues that the World Wide Web "*demands novel and specialised programming techniques*". He identifies major issues for locations, mobility and administrative domains. In a network there are two kinds of locations: *virtual location* is the notion of location known to the users; *physical location* is the place where where the machines are kept. Administrative domains, in general, manage both kinds of locations.

Both physical and virtual locations form separate networks, which might not coincide. Cardelli claims that, unlike traditional local area networks, both virtual and physical locations can move; thus models for the Internet have to represent change in the topology of the network throughout computation.

Moreover, the Internet imposes a new concept of *mobility*. Over the World-Wide Web, code moves around, as for example when downloading software. This poses, of course, serious security issues which have been widely exploited by different communities. Technological devices, however, such as lap-tops, PDAs, or mobile phones connected to a web-service, are another form of mobility. People like to move around with their devices, while they may want to connect to and disconnected from networks. However, for obvious security reasons, most organisations do not allow simply anyone to connect. Firewalls surround and protect domains, and might refuse connection according to a certain policy. Connecting, then, boils down to gaining access to an administrative domain, which means to be granted permission to enter (possibly to exit).

Mobility in this framework means exiting one administrative domain and entering another one. One concrete example is given by myself with my laptop. In the morning, I connect with my local provider. When I enter Imperial College with the laptop, and want to read my e-mail, I have to ask permission to gain access to Imperial's network. I then might download some software and leave to go back home, where I can again connect with my provider. At each step of my movements I need to access a particular administrative domain: that of Tiscali, my Internet provider, and the physical and virtual network of Imperial.

Such a day is represented as follows:



My LAPTOP and IMPERIAL are two physical objects. Tiscali and network are two virtual locations to which I can connect in order to download my software. I connect with my provider, here represented by a process for the creation of the connection, which goes inside TISCALI.

Then after disconnecting from Tiscali, I physically move to Imperial College, where I connect with the local network.

My laptop, and downloading software, are two different kinds of mobility; these kinds normally overlap. Cardelli argues that future models must represent this situation.

Issues related to security and mobility have already been the object of study by theoreticians. We report here two very successful calculi, although many other examples can be found.

Mobile code has been exploited by Milner [61, 58] within the π-calculus. This calculus has acquired a fundamental role in modeling concurrent systems that pass around resources that can be reused later. The name-passing paradigm, on which the π-calculus is based, has proved to be a powerful and simple framework for describing different scenarios appearing in concurrency. A remarkable result has been shown by Sangiorgi [76], who proved that sending processes, i.e. moving code around, can be faithfully represented in standard π-calculus. Thus, the π-calculus is a useful model for understanding issues related to code mobility.

Although the π-calculus remains a cornerstone within the world of process calculi, the feeling exists that explicit constructs are needed for modeling phenomena of the Internet. For the security aspect, Abadi and Gordon [1] consider the spi-calculus. This is an off-spring of the synchronous π-calculus augmented with cryptographic primitives. The spi-calculus had proven successful for verifying security protocols [1].

The spi-calculus and π-calculus are very interesting, but they do not make reasoning about authorisation and administrative domains easy. In order to fill that gap,in the last few years alternative calculi have been proposed to address some of those issues: locations, code mobility, abstract domains and security, or moving agents. We mention the Seal-calculus [88], $D\pi$ [42], Nomadic Pict [89], Join Calculus [31], and a local area network calculus [26].

The calculus of Mobile Ambients (MA) [24] has been much praised for its simplicity and ability to represent distributed mobile computation. It has been advocated [24] as a foundational calculus for representing phenomena present on the Internet. The main advantage of MA is the simple underpinning unifying concept of *ambient.* Ambients are meant to represent administrative domains; they have a tree structure, possibly containing sub-ambients; the notion of access and mobility is captured by the operational semantics, where processes equipped with the appropriate capability can freely enter or exit an ambient.

The calculus has become in a short time very popular [22, 20, 23, 2, 69, 13, 27, 25, 73, 15, 86, 43, 77] (just to cite a few articles). Moreover different dialects have been proposed, like Safe Ambients [48], Safe Ambients with Passwords [52], Boxed Ambient [14], Robust Ambients [40], and the Push and Pull Ambient Calculus [73], to name but a few.

The overall purpose of this dissertation is to deepen the understanding of the primitives of the

ambient calculi. We hope that, if one day the ambient calculus will play a rôle similar to the one the λ-calculus has played for functional languages, this work will be useful. We shall focus on two aspects: semantics and expressiveness.

On the semantics side, we argue that, for the ambient calculus, reduction semantics is a suitable tool for reasoning about computation. By reduction semantics we mean here computation represented in terms of rewriting rules, and equivalences defined on it. We deepen the understanding of reduction semantics by showing how to derive canonical equivalences starting from an equational point of view. We shall see that in this context the role of names is crucial.

On the expressiveness side, we shall see that the ambient calculus is quite expressive in the sense that the matching operator can be faithfully encoded, and leader election problems can be solved. These facts have profound consequences for the reciprocal encodability of ambients into the π-calculus. In order to carry out a thorough investigation, we consider in this dissertation two other ambient calculi: Safe Ambients (SA) and the Push and Pull Ambient Calculus (PAC). We hope that, in this way, the reader will have a clear view of the difficulties and subtleties involved when dealing with other calculi in the same family.

Milner [62] motivated the study of reduction semantics on the grounds that it is a uniform way of describing semantics for calculi that are syntactically different from each other. In fact, reduction semantics has been widely used [55, 58, 62, 76] owing to its simplicity and ability to represent uniformly process calculi such as CCS [58], first and second order name passing calculi such as π-calculus and $\mathcal{HO}$-π-calculus [55, 62, 76]. In our view, this is the most important motivation for further developing the study of reduction-based semantics. In [62] and [46] bisimulation based on reduction semantics has been proposed. Sensible bisimulation in this setting requires the quantification over contexts. This is in general a nuisance, because in order to show that two very simple processes are bisimilar, one needs to analyse all possible contexts. A direct definition of bisimulation over labelled transition systems is generally preferable. Traditionally, in concurrency, *labelled transition systems* are employed to define operational semantics [54, 61, 44, 5]. It is generally accepted that labelled transition systems facilitate proofs, since they explicitly describe the derivation of processes. Moreover, over labelled transition systems notion of bisimulation is defined which gives nice proof methods. This is true in general, although we argue that for for ambient calculi, semantics in terms of labelled transition systems does not yield any gain in elegance or understanding.

In the last few years many labelled transition systems have been proposed for ambient calculi [38, 39, 30, 53]. The author of this thesis also has made her own contribution [85, 86]. For dialects there is the work of Levi and Sangiorgi [48], who propose a labelled transition system for SA; for the dialect of ambients with passwords (SAP), there is [52]. Bisimulation is defined

in a complicated way, where quantification over contexts is necessary. This is needed in order to deal with restriction and the fact that processes appear and disappear in ambients; this is similar to the definition of bisimulation given by Sangiorgi [76] for $\mathcal{HO}$-π-calculus. Bisimulation was not a problem there since the higher order π-calculus has a natural translation into standard π-calculus. In real life one does not use a higher-order bisimulation, but rather the equivalent definition for standard π-calculus. However for MA there is no translation to a simpler calculus; thus, one has to deal with contexts as in reduction bisimulation.

We present the definition of barbed bisimulation [62] and contextual barbed bisimulation [46]. The latter has been studied from a foundational point of view; in fact [46] gives a general formulation of process semantics which induces a canonical congruence based solely on the reduction relation and equational reasoning. We shall see that this technique applies well to MA.

The ambient calculus has been advocated as a fundamental calculus, and the scientific community has been very interested in the comparison between MA and the π-calculus. The problem boils down to the existence of mutual encodings. Some dialects of the π-calculus have been encoded in ambient calculi [24, 73, 49, 92]. These encodings show that, in the ambient world, the behaviour of the asynchronous π-calculus can be simulated. This seems to imply that MA (or its dialects) is at least as expressive as the π-calculus (without choice).

A standing open problem, for long unsolved, regards the reverse result. The last part of this dissertation directly addresses this issue. The expectation of the scientific community is that basic MA cannot be encoded into π-calculus. In the past, we tried without success to define such an encoding. The reason for the failure of our attempts seems to be related to the tree structure inherent in ambients. It seems very difficult to simulate in π-calculus the fact that processes can move inside another ambient, disappear, keep computing and come back as a different process. Moreover, the possibility of destroying an ambient seems to entail a sort of broadcasting communication that is not encodable in π-calculus [28].

Expressiveness results are very difficult to deal with for various reasons. If encodings are defined, it is because at some point one sees how to simulate the semantics of one calculus into another. Still, there is a lot of work involved in arguing that the encoding is ‘good’ [65].

However, the hardest task is to show that an encoding does not exist. A good informal reason does not translate easily into a formal proof. This is a matter of ingenuity.

We approach the problem starting from the leader election problem, as dealt in distributed systems. We have taken inspiration from Palamidessi [70, 71], who established that π-calculus with mixed choice (i.e. where the summands in a choice can be a mixture of inputs or outputs) is strictly more expressive than the π-calculus with separate choice (i.e. where the summands must be all inputs or all outputs). We have established that a small fragment of MA is not encodable in the

π-calculus with separate choice. The subtle point here is that for the π-calculus case, mixed choice seems crucial for writing a program that solves the problem of electing a leader in a symmetric network. Choice however is not present as a primitive construct in the ambient world. Mobility in MA has the power of *pre-emption* –of inhibiting alternatives– even though it cannot remove alternatives completely, as can a choice operator. This pre-emptive power is enough to break the symmetry and elect a leader. Moreover, we show that in the ambient world, the crucial capability for the solution of the leader election problem in symmetric networks is the in as *subjective move*. In fact without this capability, in standard MA, the election problem cannot be solved. We complete our research by showing that a dialect of MA equipped with special primitives called *objective moves* [24] does not admit a solution to the leader election problem. Conclusions and ideas for future work complete the thesis.

Outline of this thesis

Chapter 1. Basic reduction semantics : This chapter aims to give a general introduction to reduction semantics. We shall deal with dialects of the π-calculus and dialects of the Ambient Calculus. All these calculi will share the reduction semantics framework. This chapter collects a body of general definitions for reduction semantics, with the aim of avoiding repetition of notions that are substantially the same in different calculi treated throughout this thesis. We provide definitions for *contextual equivalence, barbed bisimulation* and *contextual bisimulation.*

Chapter 2. From π to ambients: evolution of calculi : In this chapter, we will review the π-calculus [61], its syntax and reduction semantics, and then introduce some dialects, namely the asynchronous π-calculus [45, 10] and π-calculus with separate choice [78]. The basics of Mobile Ambients [21, 24] will also be reviewed, including its syntax and semantics, together with the Safe Ambients [48, 49].

Chapter 3. Encoding of the matching operator : In this chapter we shall present the encoding of the *matching operator* in MA. The matching operator was introduced in the π-calculus [61], and under 'certain conditions' it cannot be encoded into some dialect of the π-calculus. We offer a discussion on what can be considered a 'good encoding'. Finally, we shall present the encoding of the matching operator in MA and we will show that this encoding satisfies at least the same 'specific conditions' under which previous encodings in dialects of π-calculus failed. We conclude the chapter with an extended example of a router for sending e-mails. We claim that in this example the matching operator is extremely useful.

Chapter 4. The Push and Pull Ambient Calculus : We introduce a new dialect of the ambient cal-

culus: the Push and Pull Ambient Calculus. The philosophy of PAC is different from that of MA. We believe that this calculus can be useful for representing a resource-sensitive situations that might occur over the Internet. We show this in the example of a virtual learning centre; we believe this example could not have been modelled in standard MA. Cardelli and Gordon encode the asynchronous π-calculus in MA, and we present an encoding of the asynchronous π-calculus in PAC. Our encoding is well behaved, in the sense that it respects strong conditions such as operational correspondence and adequacy.

Chapter 5. From sound theories to contextual barbed bisimulation : In this chapter we investigate equivalences in the reduction semantics framework by the means of equational reasoning. This work has been inspired by Honda and Yoshida [46], who devised this method and applied it to various dialects of the π-calculus. They [46] have proposed a general formulation of process semantics which induces a canonical congruence based solely on reduction relation and equational reasoning. In fact throughout this chapter we shall develop a construction of sound theories that we prove to coincide with the contextual barbed bisimulation. The main contribution of this chapter is to show that Honda and Yoshida's framework can be non-trivially generalised to MA.

Chapter 6. Expressiveness results : In this chapter we shall compare the expressiveness of ambient calculi against different dialects of the π-calculus. The main contribution of this chapter is a separation result between MA and π-calculus, and the definition of a hierarchy of expressiveness within the family of ambient calculi. To this end we consider a standard problem in the literature of distributed algorithms [50, 4, 80]: *leader election problem in symmetric networks.* Palamidessi [71], adapts the the leader election problem to the labelled transition semantics of the π-calculus, and shows that the π-calculus with mixed choice cannot be encoded (under certain conditions) in the π-calculus with separate choice. Taking inspiration from her work, we adapt the leader election problem to the general framework of reduction semantics. In this way we are able to compare different calculi under *identical conditions.* We restate Palamidessi's results for π-calculusin the new setting, and we show that the pure version of MA without restriction and without the open capability admits symmetric electoral systems, that is, it is possible to solve the problem of electing a leader in a symmetric network. This implies that this fragment of the ambient calculus is not encodable in the π-calculus with separate choice. Moreover, we discuss which fragments of other dialects of the ambient calculus, PAC and SA, admits an electoral system. We shall present a hierarchy within MA, by showing that MAwithout the in capability *does not* admit an electoral system.

Chapter 7. Conclusion : In this chapter we summarise our work and we sketch how we envisage

future work in this area.

Chapter 1

Basic reduction semantics

This chapter aims to give a general introduction to reduction semantics. We shall deal with dialects of the π-calculus in Chapter 2, and dialects of the Ambient Calculus in Chapter 2 and in Chapter 4. All these calculi will share the reduction semantics framework. This chapter aims to collect the body of general definitions for reduction semantics, in order to avoid repetition of notions that are substantially the same in different calculi treated throughout this thesis.

1.1 *Historical remarks*

Reduction semantics is an operational semantics which aims to formalise (by means of mathematical entities) the intuitive meaning of terms of a given language. Operational semantics has also been an extensive object of study in computer science in general [41]. Proper to the concurrency setting, however, is the use of *labelled transition systems* to the end of defining operational semantics [54, 61, 44, 7, 5]. Given a set of *actions*, which describe what can be observed, each term of the language has an associated transition rule, that describes how the term evolves by using by exhibiting certain behaviour. For instance, the meaning of $a.P \xrightarrow{a} P$ is that $a.P$ exhibits the visible behaviour a and then evolves in P. Computation is represented by a special action τ, which is invisible,i.e. $P \xrightarrow{\tau} P$.

Traditionally in λ-calculus [6] (and rewriting systems), the operational semantics is defined via a notion of simple reduction relation on terms (β-reduction), formally expressed as $(\lambda(x).x)N \rightarrow N$. In this case there is no action above the arrow that represents computation. Labelled transition semantics has been used in the lambda calculus setting [36], while reduction semantics in the

concurrency setting originated with the Chemical Abstract Machine [8], and aims to represents the computation of processes with a minimal set of rules. The *reduction relation* aims to model the computational steps of a process in a given configuration. We use the word 'configuration' as opposed to process term, because not all terms are in the domain of the reduction relation. For instance in the CCS case, $a.P \mid \overline{a}.P$ would be a configuration, it falls in the domain of the reduction relation, however $a.P$ alone would not. The only possible information on the latter term is that it does not reduce alone, yet it might compute once composed with other processes. This is clearly in contrast with labelled transition semantics, where every single term can be analysed in terms of their visible (and invisible) behaviour. Configurations (and terms) are identified up to syntactical rearrangements via a relation, *structural congruence,* that can be thought of as a 'meta-machine' that reorganises terms into the right syntactical form for being reduced. Such a relation will not be the topic of this chapter; it will defined concretely for each calculus.

It is generally accepted that labelled transition systems facilitate proofs, for they explicitly describe the derivation of processes, while reduction semantics gives little information about terms that do not fall in the domain of the reduction relation, as pointed out earlier. However, there is no uniform way to derive labelled transition semantics. For many calculi, such as CSP [44], CCS [54], and ACP [7], labelled transition systems are canonical. However, in the case of the π-calculus there are at least three different labelled transition systems: *early*, *late* [61] and *symbolic* [47]. A formal method for deriving a labelled transition system in a given language is an active topic of research; for an exhaustive discussion we refer to [59, 60, 79].

Milner [62] motivated the study of reduction semantics on the grounds that it is an uniform way of describing semantics for calculi that are indeed syntactically different from each other. In fact, reduction semantics has been widely used [55, 58, 62, 76, 21, 88] for its simplicity and ability to represent uniformly simple process calculi such as CCS [58], first and second-order name passing-calculi such as the π-calculus and $\mathcal{HO}$-π-calculus [55, 62, 76], and more complex calculi such as the Seal Calculus [88] and the Ambient Calculus [21]. In our view, this is the most important motivation for further developing the study of reduction-based semantics.

In concurrency, representing computation is not the final goal. It is certainly interesting to represent infinite behaviour; however investigating the conditions for two terms to be regarded as equivalent leads great part of the research of this thesis. This issue has led to the definition of different equivalences with the help of labelled transition systems [83]; however, also in reduction semantics much research has been carried out [32, 62]. Equivalences would be too weak if defined on reduction semantics alone [62]; in order to obtain a sensible equivalence, the notion of *observable* is introduced, and in many cases contexts play a crucial rôle. The choice of observable is delicate, since it influences the power of observability: thus it might yield different equivalences.

We shall be interested in congruences, i.e. equivalences that are preserved by all contexts. Contexts are simply terms with a hole, that can be derived from the grammar of a language. Thus, observable predicates, and reduction relations and contexts are the necessary ingredients to define equivalences. In this thesis we take the view that a calculus is identified with:

- a set of processes;
- a reduction relation;
- an observational predicate.

These three notions are sufficient to define sensible equivalences. We are going to deal with contextual equivalence, barbed bisimulation and contextual bisimulation.

In the next sections we are going to define a general notion of observational predicate, *barb*; a generic notion of reduction relation; and a generic notion of context over a possible grammar. We then provide the definitions for *contextual equivalence, barbed bisimulation* and *contextual bisimulation.*

1.2 *Basics*

First of all, we assume the existence of a set of names $\mathcal{N}$: the variables $a, b, m, n \ldots$ range over it. Names are meant to be atomic, and they are a useful abstraction to represent objects that in real life we do not want to view as separated, such as identifiers, sequences of bits, etc. By a *nominal calculus* we mean a calculus whose definition of processes makes use of names [37].

Assume the existence of a set of processes for a language $\mathcal{P}r$, defined by a BNF grammar:

$$P \ := \ \ldots$$

Bound and free names (or variables) are standard concepts in the $\lambda-$calculus [6]. In general, some operators present in the syntax are *binding*, in the sense that names that fall within their *scope* are called *bound*, and processes that differ in bound variables only are considered identical. Names that are not bound in a process are called *free*. These concepts will be explicitly defined in each concrete syntax considered later in this thesis.

Context is a common notion that comes with syntax. Informally, contexts are terms with a hole $[\,]$, and, generally speaking, they are not identified up to bound variables. A context might contain one or more holes, depending on the definition of the concrete syntax. We write $\mathcal{C}$ for context, and unless otherwise specified, a context is defined over the full grammar of processes. Assuming that P is a process and $\mathcal{C}[\,]$ a context, we write $\mathcal{C}[P]$ as the placement of P in the hole of the context. Free names of P might become bound after this operation.

The computational steps for a given language can be captured by a simple relation over the set of processes; this kind of relation is usually called *reduction relation.*

DEFINITION 1.2.1 A *reduction relation* is a binary relation over the set of processes $\longrightarrow \subseteq \mathcal{P}r \times \mathcal{P}r$.

- We write $\longrightarrow^{=}$ for the reflexive closure of $\longrightarrow$.
- We write $\longrightarrow\!\!\!\!\!\rightarrow$ for the reflexive and transitive closure of $\longrightarrow$.

While the reduction relation captures the notion of computation, one could argue that computation as such is an invisible event, the τ action of labelled transition systems for calculi such as CCS. Concurrent calculi aim to capture infinite computation with finite representation. Still, we distinguish programs on the basis of their behaviour; hence we think of behaviour of programs as making visible effects to a possible observer.

In the framework of reduction semantics for nominal calculi, free names seem to be an obvious choice to represent the visible effect of a given process. To the end of modeling visible behaviour of programs, an *observational relation* is defined between processes and names. This relation might be defined differently in various settings. Throughout this thesis, we will use both the words 'observable' as well as 'barbs' for names in the observational relation.

DEFINITION 1.2.2 (*Barb*) Let $\downarrow$: $\mathcal{P}r \times \mathcal{N}$ be a binary relation.

- A process P *exhibits a barb* n if and only if $P \downarrow n$.
- A process P *eventually exhibits a barb* n, written $P \Downarrow n$, if and only if, for some P', $P \longrightarrow\!\!\!\!\!\rightarrow P'$ and $P' \downarrow n$.

The predicate $P \downarrow n$ is called *strong barb* and $P \Downarrow n$ is called *weak barb.*

DEFINITION 1.2.3 A *nominal calculus* defined over a set of names $\mathcal{N}$ is a triple $\mathsf{C} = \langle \mathcal{P}r, \longrightarrow, \downarrow \rangle$ where:

i) $\mathcal{P}r$ is the set of process terms or language given by a syntax;

ii) $\longrightarrow$ is a binary relation as in Definition 1.2.1;

iii) $\downarrow$ is an observational predicate as in Definition 1.2.2.

In case (*ii*) of Definition 1.2.3, the transitive and reflexive closure of the relation $\longrightarrow$ has not been mentioned. This is indeed not necessary since it is not a particular feature of the calculus, but a definition that belongs to the algebra of relations.

In what follows, for each concrete calculus, be it the ambient calculus or the π-calculus, we will define the set of processes, the reduction relation and the observational predicate; however, there is a set of equivalences that will remain the same. We give the definitions in the following sections, to avoid future repetition.

1.3 *Reduction equivalences*

The aim of equivalences is to define precisely when two syntactically different terms in a calculus can be considered to be semantically the same. Different requirements induce a variety of equivalences; for a small overview on different equivalences on labelled transition systems the reader may look at [83].

In the following section we present standard and well-known equivalences [32] in concurrency that dispense with the labelled transition system approach in favour of reduction semantics. Far from being exhaustive, our presentation aims merely to describe well-known equivalences, namely: *contextual equivalence, barbed bisimulation* and *contextual bisimulation.*

1.3.1 *Contextual equivalence*

Contextual equivalence originated in the setting of the lambda-calculus [64]. It is also known as testing equivalence within the De Nicola-Hennessy framework [68] and it has been studied in [9]. The simple idea that underpins this definition is that two processes are indistinguishable if they exhibit the same behaviour under the same set of tests.

This notion is quite realistic. To give a simple example, in a sequential setting, assume we have two programs P and Q, which are implementations of addition. Maybe, syntactically, P and Q are different; however, one could not distinguish the two on the basis of their outputs. Ideally, if we could test them on the natural numbers, we would deem them to be equal.

In the previous section, we have defined observables of processes, the barbs. Tests here are represented by all possible contexts. Thus, contextual equivalence aims to identify processes that exhibit the same barbs in all contexts.

DEFINITION 1.3.1 A relation $\mathcal{R} \subseteq \mathcal{P}r \times \mathcal{P}r$ is a congruence, if, for all contexts $\mathcal{C}$, $P\mathcal{R}Q$ implies $\mathcal{C}[P]\mathcal{R}\mathcal{C}[Q]$.

DEFINITION 1.3.2 (*Contextual equivalence*) Let $\mathsf{C} = \langle \mathcal{P}r, \longrightarrow, \downarrow \rangle$ be a generic calculus. Two processes $P, Q \in \mathcal{P}r$ are said *contextual equivalent*, $P \simeq Q$, if and only if, for all names n and all contexts $\mathcal{C}$, $\mathcal{C}[P] \Downarrow n$ if and only if $\mathcal{C}[Q] \Downarrow n$.

The following proposition is well known; hence we omit the proofs.

PROPOSITION 1.3.3 *i*) $\simeq$ over C is an equivalence relation over $\mathcal{P}r$.

ii) $\simeq$ over C is a congruence.

1.3.2 *Barbed bisimulation*

Barbed bisimulation [62] was originally defined for CCS and the π-calculus, with the aimed of applying co-inductive methods to calculi without using labelled transition systems. The original idea was to overcome the different definitions of labelled bisimulation in the π-calculus due to the presence of more than one definition of labelled transition system [61, 57, 76]. Barbed bisimulation proved successful for the uniform definition of bisimulation in $\mathcal{HO}\pi$ (higher order π-calculus). Bisimulation can be seen as a game between two machines: every time the first machine displays a barb, the second machine has a computation that will display the same barb, and vice-versa.

In the following definition we have adapted the idea presented in [62].

DEFINITION 1.3.4 (*Barbed Bisimulation*) Let $\mathsf{C} = \langle \mathcal{P}r, \longrightarrow, \downarrow \rangle$ be a generic calculus. A symmetric relation $\mathcal{S} \subseteq \mathcal{P}r \times \mathcal{P}r$ is a *weak barbed bisimulation* over C, if $P \mathcal{S} Q$ implies:

- for all barbs n, if $P \downarrow n$ then $Q \Downarrow n$;
- whenever $P \longrightarrow P'$ then there exists a Q' such that $Q \longrightarrow\!\!\!\!\!\rightarrow Q'$ and $P' \mathcal{S} Q'$.

DEFINITION 1.3.5 Two processes $P,\ Q$ are said to be *weakly barbed bisimilar over* C, written $P \dot{\approx} Q$, if and only if $P \mathcal{S} Q$ for some weak barbed bisimulation $\mathcal{S}$.

The following proposition establishes well-known properties of barbed bisimilarity; thus they are presented here without proof.

PROPOSITION 1.3.6 *i*) $\dot{\approx}$ is the largest barbed bisimulation.

ii) $\dot{\approx}$ is an equivalence relation.

In general barbed bisimilarity $\dot{\approx}$ is too large, in the sense that it equates too many processes (for specific examples, we refer the reader to Chapter 2 where concrete semantics is introduced). Also this relation is not a congruence. In order to obtain a sensible relation, we rely on the discriminating power of contexts, which yield the notion of congruence.

DEFINITION 1.3.7 (*Barbed Congruence*) Let $\mathsf{C} = \langle \mathcal{P}r, \longrightarrow, \downarrow \rangle$ be a generic calculus. Two processes $P,\ Q$ are said to be *barbed congruent,* written $P \cong^c Q$, over C if and only if $\mathcal{C}[P] \dot{\approx} \mathcal{C}[Q]$

for all contexts $\mathcal{C}$.

PROPOSITION 1.3.8 $\cong^c \subseteq \dot{\approx}$.

1.3.3 *Contextual barbed bisimulation*

As seen above, barbed congruence is defined in two steps, first of all by defining a barbed equivalence, and then by taking the largest congruence that is contained in such a relation.

There is another way of defining bisimulation within the reduction semantics framework which is by taking immediately the largest congruence. This definition was given by Honda and Yoshida [46] for the asynchronous π-calculus, and also by Montanari and Sassone for CCS [63]. In Chapter 5 will see how to obtain this relation in a different way. For the moment, we report here the operational definition adapted from the π-calculus.

DEFINITION 1.3.9 (*Contextual Barbed Bisimulation*) Let $\mathsf{C} = \langle \mathcal{P}r, \longrightarrow, \downarrow \rangle$ be a generic calculus. A symmetric relation $\mathcal{S} \subseteq \mathcal{P}r \times \mathcal{P}r$ over C is a *contextual barbed bisimulation* if $P \mathcal{S} Q$ implies:

- for all n, if $P \downarrow n$ then $Q \Downarrow n$;
- for any context $\mathcal{C}$, whenever $\mathcal{C}[P] \longrightarrow P'$ then there exists a Q' such that $\mathcal{C}[Q] \longrightarrow\!\!\!\!\!\rightarrow Q'$ and $P' \mathcal{S} Q'$.

In the previous definition, the notion of context is introduced in the definition of bisimulation.

REMARK 1.3.10 In this thesis, we will be mostly dealing with bisimulation; however the non-symmetric relation as defined above is simply called *weak contextual simulation.*

DEFINITION 1.3.11 Two processes P, Q are said to be *contextual barbed bisimilar* over C, written $P \approx Q$, if and only if $P \mathcal{S} Q$ for some contextual bisimulation $\mathcal{S}$.

The following proposition establishes well-known properties of contextual bisimilarity: thus they are presented here without proof.

PROPOSITION 1.3.12 *i*) $\approx$ is an equivalence relation.

ii) $\approx$ is a congruence.

iii) $\approx$ is the largest contextual bisimulation.

The following theorem is well known in the literature and straightforward, thus we omit the proof.

It establishes a hierarchy on the equivalences defined above.

THEOREM 1.3.13 $\approx \subseteq \cong^c \subseteq \simeq$.

Perhaps slightly differently from standard literature, we called here the equivalence relation defined in [46] (Definition1.3.9) a *contextual barbed bisimulation.* The reason for using this name is to avoid confusion with the classical name of barbed bisimulation as defined by Milner and Sangiorgi [62] (Definition 1.3.4)

1.3.4 *Other useful equivalences*

Finally, we introduce a relation that aims to capture the idea that one process is just an implementation of another. In other words, two processes differ only because one of them makes more computation steps. This equivalence is known in semantics with labelled transition systems as *expansion.* We adapt this framework from [54, 67].

DEFINITION 1.3.14 (*Strict Simulation*) Let $\mathsf{C} = \langle \mathcal{P}r, \longrightarrow, \downarrow \rangle$ be a generic calculus. A relation $\mathcal{S} \subseteq \mathcal{P}r \times \mathcal{P}r$ is a *strict weak simulation* over C, if $P \mathcal{S} Q$ implies:

- for all n, if $P \downarrow_n$ then $Q \downarrow_n$;
- for any context $\mathcal{C}$, whenever $\mathcal{C}[P] \longrightarrow P'$ there exists a Q such that $\mathcal{C}[Q] \longrightarrow^{=} Q'$ and $P' \mathcal{S} Q'$.

DEFINITION 1.3.15 *i*) A relation $\mathcal{E} \subseteq \mathcal{P}r \times \mathcal{P}r$ over C is an *expansion* if $\mathcal{E}$ is a weak contextual simulation (Remark 1.3.10) and $\mathcal{E}^{-1}$ is a strict weak simulation.

ii) Process P expands Q, written $P \lesssim Q$, if $P \mathcal{E} Q$ for some expansion $\mathcal{E}$.

The following propositions are well-known and straightforward.

PROPOSITION 1.3.16 $\lesssim \subseteq \approx$.

Proof. Consider the following relation: $\mathcal{S} = \lesssim \cup \lesssim^{-1}$. Clearly $\mathcal{S}$ is symmetric and $\mathcal{S} \subseteq \approx$.

COROLLARY 1.3.17 *i*) $\mathsf{id} \subseteq \lesssim$.

ii) $\lesssim$ is reflexive and transitive.

In the remainder of this thesis, we will drop the specification 'over C' when talking about equivalences, whenever there is not possibility of confusion.

Chapter 2

From π to ambients: evolution of calculi

In this chapter, we will review the π-calculus [61], its syntax and reduction semantics, and then we shall introduce some dialects, namely the asynchronous π-calculus [45, 10] and the π-calculus with separate choice [78]. The π-calculus will be important in Chapter 4 and Chapter 6. We assume the reader is familiar with the π-calculus; thus the exposition will be brief. The basics of Mobile Ambients (MA) [21, 24] will also be reviewed (its syntax and semantics) together with Safe Ambients (SA) [48, 49].

2.1 *The π-calculus*

The π-calculus was originally introduced [61] with the aim of representing systems whose topology changes during computation. Communication involves two parties, i.e. processes, and a common line of communication: a channel. Thus, in order for communication to happen, two processes need to share a channel. The novelty of the π-calculus is that links can be transferred from process to process and reused for later computation. In this setting, links, names or channels are identified by a unique atomic entity,a *name,* which is used in communication for both channels and messages.

With respect to previous process calculi, such as CSP [44], CCS [54], ACP [5], etc, the π-calculus pioneered the notions of both *mobility* and *scope extrusion,* in such a way that they will remain an intellectual legacy for future generations of calculi. In the π-calculus *mobility* is represented by passing names around. Previously, in CHOCS [81, 82] processes were transferred between processes. However, the underpinning semantics was very complicated. The π-calculus

is equipped with a simple and elegant semantics. *Scope extrusion* is a phenomenon proper to the π-calculus. It refers to the ability to pass around private resources, a feature not present in previous calculi. For instance, in CCS (and also in other calculi) it was possible to make a channel private to some processes in such a way that there would be no interference. Once a name was made private, it could not be used by other processes. In the π-calculus it is possible to make a name private and to pass it to another process, which might use it as channel. Hence the new process has got a new private resource that was not available before, and that can be used for later communication.

2.1.1 *The syntax of the π-calculus*

We first review the basics for the standard π-calculus with mixed choice $\pi^{=}_{m}$ (with the matching operator) [61, 58]. We refer to the *monadic* π-calculus when each channel sends and receives one name only, i.e. each channel has arity one. Otherwise, the calculus is called *polyadic.* The polyadic π-calculus can be encoded into the monadic one [55, 57]. There are different encodings of the polyadic into the monadic π-calculus; each of them respects different semantic properties. We refer the reader to [55, 57, 90, 91].

We will be concerned with the monadic π-calculus for simplicity. This will avoid introducing the discipline of types present in the polyadic π-calculus [55, 57, 76], which guarantees that there are no run-time errors, such as mismatching of arities of channels.

We assume an infinite set of names $\mathcal{N}$ ranged over by $m, n, q, y, u, x, \ldots$.

DEFINITION 2.1.1 The set of process terms of the π-calculus $\mathcal{P}_{\pi^{=}}$ is given by the following syntax:

$$
\begin{array}{rcll}
P, Q & ::= & \mathbf{0} & \textit{nil} \\
 & | & \sum_{i \in I} \alpha_i.P & \textit{choice}, \\
 & | & P \mid Q & \textit{composition} \\
 & | & [n=m]P & \textit{matching} \\
 & | & (\nu x)P & \text{restriction} \\
 & | & !P & \text{replication}
\end{array}
$$

where I is a finite set. The *prefixes* of processes ranged over by α are defined by the following syntax:

$$\alpha \quad ::= \quad m(z) \;\mid\; \overline{m}\langle q \rangle$$

The important primitives in the syntax above are the *input* and the *output.* Below their meaning is explained informally.

$m(z).P$: *input prefix,* the process is waiting for an input on channel m, before continuing with

P. Notice that z is a bound name in P.

$\overline{m}\langle q\rangle.P$: *output prefix*, the process sends the name q on the channel m and then behaves like P.

The *summation* $\sum_{i\in I}\alpha_i.P_i$ represents a finite choice among the different processes $\alpha_i.P$. This operator is also called *mixed choice*, since both input and output prefixes can be present at the same time ,as in the following: $\overline{m}\langle q\rangle.P + s(x).Q$. The symbol **0**, called *nil*, is the inactive process. Commonly in the π-calculus, **0** is an abbreviation for the empty choice. Although redundant, we introduce it here as a primitive for uniformity with the syntax of other calculi *Replication* $!P$ simulates recursion by spinning off copies of P. *Parallel composition* of two processes $P \mid Q$ represents P and Q computing independently from each other. *Restriction* $(\nu n)P$ creates a new name n in P, which is bound. *Matching* $[n=m]P$ behaves like P if and only if the two free names m, n are the same, otherwise the process is deadlocked.

NOTATION 2.1.2 In the rest of the thesis, we shall feel free to omit trailing **0**s. Thus we write α instead of $\alpha.\mathbf{0}$. We shall write $(\nu n_1 \ldots n_k)P$ instead of $(\nu n_1)\ldots(\nu n_k)P$, and sometimes we write $\widetilde{x}$ for $x_1 \ldots x_k$, when k is irrelevant or clear from the context.

The notion of *free names*, *fn*(P), and *bound names*, *bn*(P), of a term P is standard taking into account that the only binding operators are input prefix and restriction. For the sake of completeness we give the definition of free names in Definition 2.1.4. Of course, bound names are the names that are not free.

REMARK 2.1.3 We follow Sangiorgi and Walker [78] in using a convention in order to avoid that free names get bound during substitution. When considering a collection of processes and substitution, we assume that bound names of processes are chosen to be different from their free names and from the names in the substitution.

DEFINITION 2.1.4 The set of *free names* *fn*(P) of a process term P is defined as follows:

$$
\begin{aligned}
\mathit{fn}(\mathbf{0}) &\stackrel{def}{=} \emptyset \\
\mathit{fn}(\overline{m}\langle z\rangle.P) &\stackrel{def}{=} \{m, z\} \cup \mathit{fn}(P) \\
\mathit{fn}(m(w).P) &\stackrel{def}{=} \{m\} \cup (\mathit{fn}(P) - \{w\}) \\
\mathit{fn}(\textstyle\sum_{i\in I}\alpha_i.P) &\stackrel{def}{=} \bigcup_{i\in I} \mathit{fn}(\alpha_i.P_i) \\
\mathit{fn}(P \mid Q) &\stackrel{def}{=} \mathit{fn}(P) \cup \mathit{fn}(Q) \\
\mathit{fn}((\nu x)P) &\stackrel{def}{=} \mathit{fn}(P) - \{x\} \\
\mathit{fn}([n=m]P) &\stackrel{def}{=} \{n\} \cup \{m\} \cup \mathit{fn}(P) \\
\mathit{fn}(!P) &\stackrel{def}{=} \mathit{fn}(P)
\end{aligned}
$$

Although by looking at the definition of free names it is clear that both input and restriction are binding operators, there is a crucial difference between them. In $m(z).P$, z is a place-holder for a future instantiation of the name. This is very similar to the λ-calculus. Some authors, for example [46], emphasise this difference by considering two syntactic objects: names and variables. Variables are place-holders for a names. In $(\nu n)P$, n is a private name in P, that is not visible, or in other words, it cannot be used outside P. The definition of α-convertibility, which aims to capture the equivalence between terms that differ in their bound names only, will be used throughout this thesis. Hence, in contrast to other treatments of the π-calculus such as [58, 72], α-conversion is going to be performed silently. In general, we conventionally assume that free and bound names are different; hence in general we can safely assume that $bn(P) \cap fn(P) = \emptyset$.

DEFINITION 2.1.5 (*Substitution*) A *substitution* σ is a function $\sigma : \mathcal{N} \to \mathcal{N}$ that is the identity except on a finite subset.

Given a substitution σ and a process P, the application of the substitution, $\sigma(P)$, is defined as follows:

$$\begin{aligned}
\sigma(\mathbf{0}) &\stackrel{def}{=} \mathbf{0} \\
\sigma(\alpha.P) &\stackrel{def}{=} \sigma(\alpha).\sigma(P) \\
\sigma(\textstyle\sum_{i\in I}\alpha_i.P) &\stackrel{def}{=} \textstyle\sum_{i\in I}\sigma(\alpha_i.P) \\
\sigma(P \mid Q) &\stackrel{def}{=} \sigma(P) \mid \sigma(Q) \\
\sigma([n{=}m]P) &\stackrel{def}{=} [\sigma(n){=}\sigma(m)]\sigma(P) \\
\sigma((\nu n)P) &\stackrel{def}{=} (\nu n)\sigma(P) \\
\sigma(!P) &\stackrel{def}{=} !\sigma(P)
\end{aligned}$$

We write $\{q/n\}$ to say that every free occurrence of n is substituted by q.

REMARK 2.1.6 In the original version of the calculus [61] the definition of syntax of processes was slightly different. Instead of replication $!P$ an explicit form of recursive operator as *agent identifier* $A(x_1 \ldots x_m)$ was used, which is as expressive as the explicit recursive operator $rec(X).P$. It has been proven that in the π-calculus replication and recursion are equivalent [55, 78]. Thus the grammar given above has the same expressive power as the original calculus.

In some presentations [72] of the π-calculus a *mismatching operator* $[m \neq n]P$ is introduced, which behaves like P if and only if the name n is different from m. We will see throughout the development of this thesis that matching operator plays a crucial rôle (Chapter 3) because it can be encoded into the Ambient Calculus, unlike the π-calculus. This difference will be discussed in Chapter 3, and will yield some expressiveness results. For mismatching, the situation is slightly different. In fact, while it is possible to show that the mismatching operator cannot be encoded in the π-calculus, with a proof similar to that of Theorem 3.2.4, yet no encoding has been provided

for this operator. Thus in this dissertation, mismatching will be object of marginal consideration.

In addition, many presentations of the π-calculus introduce the silent prefix τ [78, 72]. In general this aims to make the comparison with labelled semantics and equational theory smoother. Since we are concerned here with reduction semantics only, we omit the notation $\tau.P$, which can be also thought of as a simplification of the process $(\nu a)(\nu r)(a(x) \mid \overline{a}\langle r\rangle).P$ with $r, x, a \notin fn(P)$.

2.1.2 *Operational semantics*

Steps of computation are performed when an input and an output are in parallel, in which case the name of the output gets substituted for all the occurrences of the bound name of the input. This will be clearer in the formal semantics via the reduction relation.

The π-calculus is equipped with an operational semantics in terms of both reduction semantics and labelled transition systems. Here we shall deal with the reduction semantics only.

The reduction semantics is usually defined in two steps, the definition of *structural congruence* which rearranges terms into contiguous positions, and the notion of *reduction relation* that captures computation on terms.

Structural congruence, written as $\equiv_{\pi^=}$, identifies processes that we do not want to differentiate for any semantic reason, or in other words, it allows syntactical rearrangement of contiguous terms not in the syntactical form for being reduced.

Structural equivalence is a *process equivalence* which means that it is an equivalence relation preserved by all contexts.

NOTATION 2.1.7 We reserve η for a bijection on I; we write $\sum_{\eta(i)\in I}$ for permutation on the sub-processes in the choice operator.

DEFINITION 2.1.8 The *structural congruence* $\equiv_{\pi^=}$ is the smallest congruence over $\mathcal{P}_{\pi^=}$ that satisfies the following equations:

$$
\begin{array}{rcl}
P \mid \mathbf{0} & \equiv_{\pi^=} & P \\
P \mid Q & \equiv_{\pi^=} & Q \mid P \\
(P \mid Q) \mid R & \equiv_{\pi^=} & Q \mid (P \mid R) \\
(\nu y)\mathbf{0} & \equiv_{\pi^=} & \mathbf{0} \\
(\nu x)(\nu y)P & \equiv_{\pi^=} & (\nu y)(\nu x)P \\
(\nu y)(P \mid Q) & \equiv_{\pi^=} & P \mid (\nu y)Q \quad \text{if } y \notin fn(P) \\
[n=n]P & \equiv_{\pi^=} & P \\
!P & \equiv_{\pi^=} & P \mid !P \\
\sum_{i\in I}\alpha_i.P_i & \equiv_{\pi^=} & \sum_{\eta(i)\in I}\alpha_{\eta(i)}.P_{\eta(i)}
\end{array}
$$

The computational step is captured by a rewriting rule from terms to term, as defined below.

NOTATION 2.1.9 We reserve S, T for summation, which means that in Definition 2.1.8

$$\overline{n}\langle q\rangle.P' + S \quad \equiv_{\pi^=} \quad \sum_{i\in I}\alpha.Q_i$$

for some suitable I and Q. We have to observe that in the case of the π-calculus with mixed choice S could contain both input and output.

DEFINITION 2.1.10 The *reduction relation* over $\mathcal{P}_{\pi^=}$, written $\longrightarrow_\pi \subseteq \mathcal{P}_{\pi^=} \times \mathcal{P}_{\pi^=}$ is the smallest relation satisfying the following rules:

$$(m(y).P + T) \mid (\overline{m}\langle z\rangle.Q + S) \longrightarrow_\pi P\{z/y\} \mid Q \quad \text{RED COMM}$$

$$\frac{P \longrightarrow_\pi P'}{P|Q \longrightarrow_\pi P'|Q} \quad \text{RED PAR}$$

$$\frac{P \longrightarrow_\pi P'}{(\nu x)P \longrightarrow_\pi (\nu x)P'} \quad \text{RED RESTR}$$

$$\frac{P \equiv_{\pi^=} Q \longrightarrow_\pi Q' \equiv_{\pi^=} P'}{P \longrightarrow_\pi P'} \quad \text{RED CONG}$$

The following barbs represent the most basic observations we can make of processes.

DEFINITION 2.1.11
- A process P *exhibits an output barb* $\overline{n}$, written $P \downarrow^o \overline{n}$, if and only if, for some P', P'', S, $P \equiv_{\pi^=} (\nu p_1 \ldots p_n)((\overline{n}\langle q\rangle.P' + S) \mid P'')$ with $n \notin \{p_1, \ldots, p_n\}$.
- A process P *exhibits an input barb* n, written $P \downarrow^i n$, if and only if, for some P', P'', S, $P \equiv_{\pi^=} (\nu p_1 \ldots p_n)((n(q).P' + S) \mid P'')$ with $n \notin \{p_1, \ldots, p_n\}$.

REMARK 2.1.12 Strictly speaking, the presence of these two barbs conflicts with our definition of calculus in the general framework of Chapter 1. There we assumed that a calculus has one barb only, instead the π-calculus with mixed choice has two predicates. We shall see that this formulation is indeed adequate for later development. However, for the time being, by abuse of notation we write the π-calculus with mixed choice as the triple $\pi^=_m \stackrel{def}{=} \langle \mathcal{P}_{\pi^=}, \longrightarrow_\pi, \{\downarrow^i, \downarrow^o\}\rangle$ considering a set of barbs instead of one barb only. We assume that the definition of equivalences will be modified accordingly. In the future, we shall omit the subscripts, and simply write $\downarrow$ when it is clear which barb we are referring to. In this thesis, we are not going to use equivalences for the π-calculus with mixed choice. Those are presented here for the sake of completeness.

Clearly, the notions of context can be derived from Chapter 1. The notion of weak barb $P \Downarrow n$ is derivable from Definition 1.2.2.

DEFINITION 2.1.13
- We write $\simeq_{\pi^=}$ for contextual equivalence defined over $\pi_m^=$ as in Definition 1.3.2
- We write $\dot{\approx}_{\pi^=}$ for barbed bisimilarity defined over $\pi_m^=$ as in Definition 1.3.4.
- We write $\cong^c_{\pi^=}$ for barbed congruence defined over $\pi_m^=$ as in Definition 1.3.7.

PROPOSITION 2.1.14 *i)* $\cong^c_{\pi^=} \subset \simeq_{\pi^=}$.

ii) $\simeq_{\pi^=} \not\subset \cong^c_{\pi^=}$.

Proof. For (*ii*), we briefly show a counter example; (*i*).

$$\begin{array}{lll} a(x).(b(y)+c(y)) & \simeq_{\pi^=} & a(x).b(y)+a(x).c(y) \\ a(x).(b(y)+c(y)) & \not\cong^c_{\pi^=} & a(x).b(y)+a(x).c(y). \end{array}$$

For calculi with choice, contextual barbed bisimulation is not adequate since, as explained in [78] (page 117), it would not be a sensible equivalence; observe that $\alpha.\tau.P \not\cong^c_{\pi^=} \alpha.P$, where τ is the silent prefix as in Remark 2.1.6.

2.1.2.1 *The matching operator*

In the previous section, we have presented the grammar of the π-calculus. Although the matching operator was originally part of the syntax of the π-calculus, it is often the case that this operator is not mentioned. In this case we talk about the π-calculus with mixed choice but not matching.

NOTATION 2.1.15 For this calculus, we write $\mathcal{P}r_\pi$ for the set of processes obtained as in Definition 2.1.1 where the matching is not included. For such a calculus, structural congruence does not include the law $[n=n]P \equiv_{\pi^=} P$, and we write simply $\equiv_\pi$. It is quite obvious that $\equiv_\pi \subset \equiv_{\pi^=}$. Hence, the π-calculus with mixed choice but without matching π_m is defined as

$$\pi_m \stackrel{def}{=} \langle \mathcal{P}r_\pi, \longrightarrow_\pi, \{\downarrow^i, \downarrow^o\} \rangle$$

. For this calculus the set of contexts is clearly smaller and we write $\simeq_\pi$ for contextual equivalence defined over π_m as in Definition 1.3.2 and $\cong^c_\pi$ for the barbed congruence as in Definition 1.3.7.

The matching operator is optional in other calculi, as we will see later. In [17] it has been proved that under certain conditions, the matching operator cannot be encoded. We will discuss this matter in Chapter 3.

2.1.3 *The asynchronous π-calculus*

The asynchronous π-calculus was independently introduced by Honda and Tokoro [45] and Boudol [10]. The grammar differs from the original π-calculus [61] for two main features: the lack of choice operator and the output operator is asynchronous, i.e there is no continuation on this primitive. The motivation behind latter choice is faithfulness to the reality of distributed systems, where full synchrony is not achievable. In this dialect of π-calculus the matching operator is considered optional.

The set of processes $\mathcal{P}r_{\pi_a^=}$ is defined by the following grammar:

$$P \quad ::= \quad \mathbf{0} \mid \overline{m}\langle n\rangle \mid m(x).P \mid !P \mid P|Q \mid (\nu n)P \mid [n=m]P$$

The asynchronous π-calculus differs from its progenitor in its for the syntax but also for the barbs. In this setting, only observation of output is allowed ($P \downarrow^o \overline{n}$ as in Definition 2.1.11). This observable, the output, is chosen under the assumption that in asynchronous systems, sending messages enables computation; hence this is the action that triggers a change in the system. Thus, the asynchronous π-calculus with matching is defined by the following triple:

$$\pi_a^= \quad \stackrel{def}{=} \quad \langle \mathcal{P}r_{\pi_a^=}, \longrightarrow_\pi, \downarrow^o \rangle$$

As above we get the following equivalences on π_a.

DEFINITION 2.1.16
- $\simeq_{\pi_a^=}$ is the contextual equivalence defined over $\pi_a^=$ as in Definition 1.3.2
- $\cong^c_{\pi_a^=}$ for the barbed congruence defined over $\pi_a^=$ as in Definition 1.3.7.
- $\approx_{\pi_a^=}$ is the contextual bisimilarity defined over $\pi_a^=$ as in Definition 1.3.2.

The asynchronous π-calculus has a particular hierarchy of equivalences, different from Theorem 1.3.13, as written in the following proposition.

PROPOSITION 2.1.17 $\cong^c_{\pi_a^=} \; = \; \approx_{\pi_a^=} \; \subset \simeq_{\pi_a^=}$.

Proof. By Theorem 1.3.13 it is the case that $\approx_{\pi_a^=} \subseteq \cong^c_{\pi_a^=} \subseteq \simeq_{\pi_a^=}$. It has been proved in [32] that $\cong^c_{\pi_a^=} \subset \approx_{\pi_a^=}$ thus it remains to be shown that $\simeq_{\pi_a^=} \not\subset \approx_{\pi_a^=}$. We write $P \oplus Q \stackrel{def}{=} (\nu ar)(a(x).P \mid a(x).Q \mid \overline{a}\langle r\rangle)$ with $r, x, a \notin fn(P, Q)$. Thus, it is not difficult to see that $(P \oplus Q) \oplus R \simeq_{\pi_a^=} P \oplus (Q \oplus R)$ but $(P \oplus Q) \oplus R \not\cong^c_{\pi_a^=} P \oplus (Q \oplus R)$. This is shown in [32].

REMARK 2.1.18 In the asynchronous π-calculus: $a(x).\overline{a}\langle x\rangle \approx_{\pi_a^=} \mathbf{0}$ [45], which would be false if one could observe inputs as well (namely $P \downarrow^i n$), since $a(x).\overline{a}\langle x\rangle \downarrow^i a$ while $\mathbf{0}$ has no barbs at all. Thus, as said in the introduction to this chapter, the choice of barbs is quite delicate, in the sense, that it might change the given equivalences.

2.1.4 *The π-calculus with separate choice*

The choice operator as described in Section 2.1 is called *mixed choice* because both input and output are allowed within the same 'choice' as in the following example $n(y).P + \overline{s}\langle t\rangle.Q$.

The π-calculus with separate choice π_s [78] is the sub-calculus of π_m where summations cannot mix input and output guards. The set of processes $\mathcal{P}r_{\pi^s}$ is given by the following grammar:

$$\alpha^I ::= m(n)$$
$$\alpha^O ::= \overline{m}\langle n\rangle$$
$$P, Q ::= \mathbf{0} \mid \textstyle\sum_{i\in I}\alpha_i^I.P_i \mid \sum_{i\in I}\alpha_i^O.P_i \mid !P \mid P|Q \mid (\nu n)P$$

One could regard π_s as having the same expressive strength as the asynchronous π-calculus [45, 10], in view of the results on encoding of separate choice [65] for the calculus without matching. We will only consider here the calculus without matching which we write as

$$\pi_s \stackrel{def}{=} \langle \mathcal{P}r_{\pi^s}, \longrightarrow_\pi, \{\downarrow^i, \downarrow^o\}\rangle.$$

2.2 *Mobile Ambients*

The aim of the Ambient Calculus [21, 24] is to model new computational phenomena over wide-areas network or Internet. In this chapter, we shall focus on two calculi: *Mobile Ambients (*MA*)* and *Safe Ambients (*SA*)*.

Since its original definition in 1998, the Ambient Calculus has spawned a number of different dialects; each of them aims at a particular application or enjoys particular useful properties. We are going to analyse the closest 'relatives' of MA and MA itself, leaving for future work the comparison with other dialects.

The different dialects that we are going to analyse have most of the operators in common, and in particular the concept of *ambient* plays a central role. Ambients are meant to represent bounded places for computation such as concrete locations, concrete domains, abstract domains, or laptop computers. Ambients move into and out of other ambients bringing along moving code, static processes and possibly other ambients.

In the following paragraph, that has been inspired by [19, 21], we will try to outline the main features of the concept of ambient. In MA the ambients are represented as follows:

- An ambient defines a perimeter, a boundary, that establishes what is inside the ambient and what is outside the ambient.
- An ambient has a *name*.
- Ambients can move around: they enter or exit other ambients.

- An ambient is a collection of *local agents*, i.e. processes, which run directly inside the ambient. $n\,[P]$ is the syntax for an ambient whose name is n with the running process P inside.
- An ambient may have other ambients inside, creating a hierarchy of nested ambients, which could be represented as a tree. Each sub-ambient has its own name and behaves as an independent ambient.
- An ambient moves with all the sub-ambients and processes inside.

2.2.1 *The syntax*

The language of MA inherits a few operators from standard process calculi: *composition, restriction* and *replication.* From the π-calculus, MA inherits the communication primitives of receiving a name, and in particular the feature of sending *private names*. The new primitives are the *ambient* and a special form of guard that goes under the name of *capability.*

We shall assume the existence of a set of names $\mathcal{N}$, the metavariables $n, m, z, s, \ldots$ range over the this set.

DEFINITION 2.2.1 The set of *process terms* of MA, $\mathcal{P}r_a$ is given by the following syntax:

$$
\begin{array}{rcll}
P, Q & ::= & \mathbf{0} & \textit{nil} \\
 & | & P \mid Q & \textit{composition} \\
 & | & (\nu n)P & \textit{restriction} \\
 & | & n\,[P] & \textit{ambient} \\
 & | & M.P & \textit{prefix} \\
 & | & !P & \textit{replication} \\
 & | & (n).P & \textit{anonymous input} \\
 & | & \langle n \rangle & \textit{anonymous output}
\end{array}
$$

where M stands for the *capabilities* defined by the following grammar:

$$
\begin{array}{rcll}
M & ::= & \mathsf{in}\, n & \textit{enter capability} \\
 & | & \mathsf{out}\, n & \textit{exit capability} \\
 & | & \mathsf{open}\, n & \textit{open capability}
\end{array}
$$

Nil, written $\mathbf{0}$, represents the inactive process, the process that does not reduce. *Ambient,* written $n\,[P]$, is composed of two parts (it is a binary operator): n is the name of the ambient, P is the active process inside. The square brackets around P indicate the perimeter of the ambient. If the ambient moves, everything inside moves with it. *Parallel composition,* written $P \mid Q$ means that P and Q are running in parallel and can compute independently from each other. *Restriction,*

written $(\nu n)P$, of the name n, makes the name private and unique to P. No other process can use this name for interacting with P. Restriction is a binder and P is its scope. Given a process P, there is a natural notion of *free names* (*fn*(P)) (Definition 2.2.2). *Replication*, written $!P$, simulates recursion by spinning off copies of P. *Action prefix* written $M.P$, represents a process where P is enabled only if the prefix M has been consumed. *Anonymous input* written $(n).P$, represents a process waiting for a name to be sent. This operator is a binder and the name n is bound in P. Unlike the π-calculus, communication happens without channels (anonymously). *Anonymous output* written $\langle n \rangle$ represents an asynchronous sending primitive. The output is not a prefix, unlike the input. We have chosen that only names can be sent. This formulation is simpler than [24], where also capabilities can be object of communication, however it is adequate for the purpose of this dissertation. Capabilities can be thought of as terms that enable the ambients to perform some actions. An ambient gains the ability to go inside another ambient whose name is n with the $\mathsf{in}\, n$ capability. An ambient gains the ability to leave a parent ambient whose name is n with the $\mathsf{out}\, n$ capability. An ambient named n can be dissolved by the means of the $\mathsf{open}\, n$ capability.

We consider here the monadic calculus, i.e. the anonymous primitives send and receive one name only, for simplicity. MA appears in the literature in both monadic [21, 39] and polyadic form, i.e. the anonymous primitive send and receive more than one name [22, 20, 37]. In the polyadic case, there is the possibility of 'run-time errors' due to the mismatch of arity of the communication primitives. a problem similar arises in the polyadic π-calculus without a sorting discipline [57]. The type system devised in [22] aims to avoid such errors.

Of course in the case of the monadic calculus, this kind of run-time error is avoided, simply because there is only one arity in the communication primitives. In conclusion, monadic MA is adequate for the purpose of the current research.

In the presentation of syntax above, we have chosen the replication operator, $!P$ [21, 22, 38, 20, 39, 37], as opposed to the recursion operator $rec(X).P$ used in [15, 25]. It is well known that standard recursion can encode replication, $[\![!P]\!]_! \stackrel{def}{=} rec(X).(P \mid X)$, while the converse might in general not be true. In π-calculus, the two operators are equivalent [57, 58];however it remains an open question whether in the case of MA replication can encode recursion [15, 25]. In general, unless otherwise specified in this thesis, we will refer to the set of processes with replication.

DEFINITION 2.2.2 The set of *free names* $fn(P)$ of a term P is defined as follows:

$$\begin{aligned}
fn(\mathbf{0}) &\stackrel{def}{=} \emptyset \\
fn(P \mid Q) &\stackrel{def}{=} fn(P) \cup fn(Q) \\
fn((\nu n)P) &\stackrel{def}{=} fn(P) - \{n\} \\
fn(n\,[P]) &\stackrel{def}{=} fn(P) \cup \{n\} \\
fn(M.P) &\stackrel{def}{=} fn(M) \cup fn(P) \\
fn(!P) &\stackrel{def}{=} fn(P) \\
fn((n).P) &\stackrel{def}{=} fn(P) - \{n\} \\
fn(\langle n \rangle) &\stackrel{def}{=} \{n\}
\end{aligned}$$

$$\begin{aligned}
fn(\mathsf{in}\, n) &\stackrel{def}{=} \{n\} \\
fn(\mathsf{out}\, n) &\stackrel{def}{=} \{n\} \\
fn(\mathsf{open}\, n) &\stackrel{def}{=} \{n\}
\end{aligned}$$

Bound names $bn(P)$ are the names that are not free in P. Substitution of bound names is called *α-conversion*. As canonical in the literature we identify terms that are α-convertible, and we will use α-conversion silently throughout this thesis. As previously in the π-calculus we assume that bound and free names in a process are distinct; thus in general $fn(P) \cap bn(P) = \emptyset$. A *substitution* is a function from names to names such that 'acts' on a finite subset of names.

DEFINITION 2.2.3 A *substitution* σ is a function $\sigma : \mathcal{N} \to \mathcal{N}$ which is the identity except on a finite subset.

Given a substitution σ and a capability M the application of the substitution, $\sigma(M)$, is defined as follows:

$$\begin{aligned}
\sigma(\mathsf{in}\, n) &\stackrel{def}{=} \mathsf{in}\, \sigma(n) \\
\sigma(\mathsf{out}\, n) &\stackrel{def}{=} \mathsf{out}\, \sigma(n) \\
\sigma(\mathsf{open}\, n) &\stackrel{def}{=} \mathsf{open}\, \sigma(n)
\end{aligned}$$

Given a substitution σ and a process P the application of the substitution, $\sigma(P)$, is defined as follows:

$$\begin{aligned}
\sigma(\mathbf{0}) &\stackrel{def}{=} \mathbf{0} \\
\sigma(P \mid Q) &\stackrel{def}{=} \sigma(P) \mid \sigma(Q) \\
\sigma((n).P) &\stackrel{def}{=} (n).\sigma(P) \\
\sigma(n\,[P]) &\stackrel{def}{=} \sigma(n)\,[\sigma(P)] \\
\sigma(M.P) &\stackrel{def}{=} \sigma(M).\sigma(P) \\
\sigma(!P) &\stackrel{def}{=} \sigma(P) \\
\sigma((\nu n)P) &\stackrel{def}{=} (\nu n)\sigma(P) \\
\sigma(\langle n \rangle) &\stackrel{def}{=} \langle \sigma(n) \rangle
\end{aligned}$$

We write $\{m/n\}$ to specify that n is replaced by m for some substitution σ i.e $\{(m, n)\} \in \sigma$.

For this definition of substitution the observation in Remark 2.1.3 applies.

NOTATION 2.2.4 Where no confusion is possible, we will use the shorthand M instead of $M.\mathbf{0}$, and $n\,[\,]$ instead of $n\,[\mathbf{0}]$. Moreover we will write $(\nu n_1 \ldots n_k)P$ instead of $(\nu n_1)\ldots(\nu n_k)P$, and sometimes we write $\widetilde{x}$ for $x_1 \ldots x_k$, when k is irrelevant or clear from the context.

2.2.2 *Operational semantics*

Operational semantics defines interaction among terms. The meaning of the computation is given by the basic movement that ambients are able to make: entering an ambient, exiting an ambient and dissolving an ambient, plus communication. Graphically the steps of computation are shown below.

ENTERING AN AMBIENT : The following diagram shows the reduction for the ambient named r, that has the ability to move into the sibling n.

$$r\boxed{\text{in } n.P \mid Q} \mid n\boxed{R} \longrightarrow_a n\boxed{r\boxed{P \mid Q} \mid R}$$

EXIT FROM AN AMBIENT : The following diagram shows the reduction for the ambient named r, that has the ability to move out of the parent n.

$$n\boxed{r\boxed{\text{out } n.P \mid Q} \mid R} \longrightarrow_a r\boxed{P \mid Q} \mid n\boxed{R}$$

OPENING AN AMBIENT : The following diagram shows the reduction for the ambient named n, that can be dissolved by a process holding the right capability.

$$\mathsf{open}\, n.P \mid n\boxed{R} \longrightarrow_a P \mid R$$

COMMUNICATION : The following diagram shows the communication between two processes:

$$\langle n \rangle \mid (x).Q \longrightarrow_a Q\{n/x\}$$

Formally, steps of computation are represented by a *reduction relation* which is defined below. *Structural congruence,* $\equiv_a$, (as it appears in the rule RED CONG below), formally introduces structural rearrangements of terms. Structurally equivalent terms have the same semantics. This relation is developed from the metaphor present in the 'Chemical Abstract Machine' [8].

A *redex* a is process that can reduce, like $m\,[\mathsf{in}\, n.P \mid Q] \mid n\,[R]$, or $n\,[m\,[\mathsf{out}\, n.P \mid Q] \mid R]$, or $\mathsf{open}\, n.Q \mid n\,[R]$, or $\langle q \rangle \mid (n).P$. This terminology is inherited from the λ-calculus. Not all processes are redexes, and some processes, such as $n\,[R] \mid \mathsf{open}\, n.Q$ could become a redex by rearranging the terms accordingly.

Structural equivalence is a *process equivalence* which means that it is an equivalence relation preserved by all operators.

DEFINITION 2.2.5 The *structural congruence relation* is the smallest congruence over $\mathcal{P}r_a$ that satisfies the following equations:

$$\begin{aligned}
P \mid \mathbf{0} &\equiv_a P \\
P \mid Q &\equiv_a Q \mid P \\
(P \mid Q) \mid R &\equiv_a Q \mid (P \mid R) \\
(\nu y)\mathbf{0} &\equiv_a \mathbf{0} \\
(\nu m)(\nu n)P &\equiv_a (\nu n)(\nu m)P \\
(\nu n)(P \mid Q) &\equiv_a P \mid (\nu n)Q \text{ if } n \notin \mathit{fn}(P) \\
(\nu m)n\,[P] &\equiv_a n\,[(\nu m)P] \text{ if } n \neq m \\
!P &\equiv_a P \mid !P \\
!\mathbf{0} &\equiv_a \mathbf{0}
\end{aligned}$$

DEFINITION 2.2.6 The *reduction relation* $\longrightarrow_a \subseteq \mathcal{P}r_a \times \mathcal{P}r_a$ is the smallest relation satisfying the following set of rules:

$$\begin{aligned}
m\,[\mathsf{in}\,n.P \mid Q] \mid n\,[R] &\longrightarrow_a n\,[m\,[P \mid Q] \mid R] && \text{RED IN} \\
n\,[m\,[\mathsf{out}\,n.P \mid Q] \mid R] &\longrightarrow_a m\,[P \mid Q] \mid n\,[R] && \text{RED OUT} \\
\mathsf{open}\,n.Q \mid n\,[R] &\longrightarrow_a Q \mid R && \text{RED OPEN} \\
\langle q \rangle \mid (n).P &\longrightarrow_a P\{q/n\} && \text{RED COMM}
\end{aligned}$$

$$\frac{P \longrightarrow_a P'}{P \mid R \longrightarrow_a P' \mid R}\ \text{RED PAR} \qquad \frac{P \longrightarrow_a P'}{(\nu n)P \longrightarrow_a (\nu n)P'}\ \text{RED RESTR}$$

$$\frac{P \longrightarrow_a P'}{n\,[P] \longrightarrow_a n\,[P']}\ \text{RED AMB} \qquad \frac{P \equiv_a P' \longrightarrow_a Q' \equiv_a Q}{P \longrightarrow_a Q}\ \text{RED CONG}$$

2.2.2.1 *Equivalences*

For MA the canonical observable is the name of an ambient at the top level. [21, 24] Thus, the predicate $P \downarrow n$ intuitively says that in P there is a top level process which is an ambient, whose unrestricted name is n.

DEFINITION 2.2.7 (*Strong barbs*) A process P *exhibits a barb* n, written as $P \downarrow n$, if and only if $P \equiv_a (\nu p_1 \ldots p_n)(n\,[P'] \mid P'')$, for some P', P'' and $n \notin \{p_1 \ldots p_n\}$.

The notion of weak barb $P \Downarrow n$ is derivable as in Definition 1.2.2. Hence, by the means of the framework described in Chapter 1 we formally define MA as:

$$\mathsf{MA} \stackrel{def}{=} \langle \mathcal{P}r_a,\ \longrightarrow_a, \downarrow \rangle$$

.We write $\mathsf{MA}^* \stackrel{def}{=} \langle \mathcal{P}r_a, \longrightarrow_a, \downarrow \rangle$ for the pure version without communication primitives. Clearly the notion of context for this calculus is given by the grammar presented in Definition 2.2.1, so finally we can present the definition of the equivalences in the spirit of Chapter 1.

The novel core of MA is represented by ambients and their movements, while anonymous communication primitives are 'borrowed' from the π-calculus, and preserve in spirit the same semantics, for example extruding a private name. However, as we will see in Section 2.2.3 (see also [24]) the fragment without communication primitives is still quite expressive. We will call the fragment without communication primitive 'pure ambient calculus' or 'pure mobile ambients'. The set of processes $\mathcal{P}r_a$ is the same as defined in Definition 2.2.1, without communication primitives.

DEFINITION 2.2.8
- We write $\simeq_a$ for contextual equivalence defined over MA as in Definition 1.3.2.
- We write $\dot{\approx}_a$ for barbed bisimilarity defined over MA as in Definition 1.3.4.
- We write $\cong_a^c$ for barbed congruence defined over MA as in Definition 1.3.7.
- We write $\approx_a$ for contextual barbed bisimilarity defined over MA as in Definition 1.3.11.

REMARK 2.2.9 We have reported here the original definition given by Cardelli and Gordon [21, 24] for the barbs. In some ways, it seems quite reasonable to observe the name of an ambient since the operator 'ambient' is present in the redexes for the pure version of the calculus. Yet, one might question such choice of barb as opposed to another one such as $(\nu p_1 \ldots p_n)(w\ [\mathsf{in}\, n.P \mid P']) \downarrow^e n$ with $n \notin \{p_1 \ldots p_n\}$.

Another way of looking at this issue is related to the equivalence relations that one tries to capture. In [21, 24] Cardelli and Gordon proved the so called *Perfect Firewall Equation* $(\mathsf{PFE})\ (\nu n) n\ [P] \simeq_a \mathbf{0}$ if $n \notin fn(P)$. The PFE can be read as: "assume that ambient named n represents a firewall, and P is the network inside; if n is secret, then nobody has access to it". In fact it behaves as the inactive process, in a weak sense, for a lot of computation can happen inside the firewall, but no behaviour can be made observable. Thus, the current barb is the minimal predicate that gives a sensible discriminating power, such that the equation that Cardelli and Gordon had in mind, PFE, is indeed in the same class of equivalence as the inactive process. Nonetheless, it remains an open question whether other barbs could have been chosen, such as the ability to enter another ambient as expressed before. Assume that one wishes to define an equivalence, that takes into account the resource of an ambient,then entering an ambient is a crucial operation, and certainly more accurate than simply observing an inert ambient. One might think of changing the observable predicate, as in $(\nu p_1 \ldots p_n)(w\ [\mathsf{in}\, n.P \mid P']) \downarrow^e n$. Then one would define a different class of equivalences. It

is clear that, even if $n \notin fn(P)$ then $(\nu n)n\,[P] \not\approx_a \mathbf{0}$; in fact if $P = \mathsf{in}\,q.P'$ then $(\nu n)n\,[P] \downarrow^e q$ while $\mathbf{0} \not\downarrow^e q$, similarly to what happened in the asynchronous π-calculus (Remark 2.1.18). Of course, similar observations would apply to other equivalences.

2.2.3 *Expressiveness*

The π-calculus [55, 57, 58, 78] and CCS [54] are Turing complete. Moreover, in the π-calculus some common data types such as lists, conditionals, trees, and objects are encodable [78]. A reasonable question to ask is to what extent such encodings are possible in MA or in some its fragments. Some of these encodings were given in the original paper [24]; others are more recent. The purpose of the section below is to give the reader a broad overview of the expressiveness of MA. More results related to expressiveness with respect to the π-calculus will be given in Chapter 6, which are the fruit of our own research. The list below is not exhaustive, but reports the most significant results.

- Mobile Ambients without communication primitives is Turing complete as shown in [24].
- Mobile Ambients without communication primitives and restriction operator, ν, is Turing complete as shown in [15, 43].
- Mobile Ambients without communication primitives and restriction operator and without the $\mathsf{open}\,n$ capability is Turing complete as shown in [51].
- Mobile Ambients without communication primitives and movements primitives (i.e. in and out) is Turing complete as shown in [15]
- Many constructions such as numeral, choice, conditional are encodable in the standard Mobile Ambients [24].
- The asynchronous π-calculus is (weakly) encodable in the calculus with communication primitives [21] and in the calculus with basic types [22].

2.3 *Safe Ambients*

Levi and Sangiorgi proposed Safe Ambients (SA) [48, 49] as a a substantial modification of MA, which retains the same computational model while improving the underpinning algebraic theory. In [48] it was argued that the basic operational semantics for MA led to the phenomenon of *grave interference*, where two or more redexes of different kind destroy each other. Let's elaborate this further. *Interference* is a common phenomenon in concurrency, in the presence of two overlapping redexes, for instance $\mathsf{open}\,m.Q \mid n\,[P] \mid \mathsf{open}\,n.R$. This process has two redexes, and two

possible reductions:

$$\begin{aligned} \mathsf{open}\, n.Q \mid n\,[P] \mid \mathsf{open}\, n.R &\longrightarrow_a Q \mid P \mid \mathsf{open}\, n.R \\ \mathsf{open}\, n.Q \mid n\,[P] \mid \mathsf{open}\, n.R &\longrightarrow_a \mathsf{open}\, n.Q \mid P \mid R. \end{aligned}$$

Observing that $\mathsf{open}\, n.Q \mid P \mid R \not\longrightarrow_a Q \mid P \mid \mathsf{open}\, n.R$ and vice-versa; we conclude that only one redex can carry out step of computation. These kinds of systems are called in term rewriting system *non-confluent.* Interferences make reasoning on programs more difficult, as pointed out in [48, 49]. However, Levi and Sangiorgi [48, 49] claim that MA presents a more aggravated form of interference; an example is given below.

$$k\,[P] \mid n\,[\mathsf{in}\, k.Q \mid m\,[\mathsf{out}\, n.K] \mid \mathsf{open}\, m]$$

In this process, the ambient m might exit n, either inside k or outside k, depending on possible reduction involving n. In general, predicting where the ambient m is going to end up would be quite difficult in a more complex system. Yet, there is no way we can guarantee that m should be out of n before n enters k, since in general an ambient cannot prevent other ambients acting upon it, once its own name is disclosed. In the original paper [48, 49], there are other cases of grave interference.

In order to overcome this problem, Levi and Sangiorgi have introduced *co-capabilities* to the ambient primitives. Co-capabilities are capabilities inside an ambient, that control the influence that other ambients have upon it. Since an ambient can be entered, exited and opened, there are three co-capabilities that determine if an ambient can be entered, exited or opened. This change induces a synchronization as well, namely in order for a reduction to occur, *both* parties need to be equipped with the right capability (resp. co-capability). The original papers [48, 49] presented a new type system that will not be considered in this thesis, since type systems are outside the scope of this work.

2.3.1 *The syntax of Safe Ambients*

In this section we introduce the primitives for Safe Ambients [48, 49]. We assume there is an infinite set of names, $\mathcal{N}_s$, ranged over by $n, m, q, r, \ldots$.

DEFINITION 2.3.1 The set of *process terms* of the safe ambients $\mathcal{P}r_s$ is given by the following

syntax:

$$
\begin{array}{rcll}
P, Q & ::= & \mathbf{0} & \textit{nil} \\
 & | & P \mid Q & \textit{composition} \\
 & | & (\nu n) P & \text{restriction} \\
 & | & n\,[P] & \textit{ambient} \\
 & | & M.P & \textit{prefix} \\
 & | & !P & \textit{replication} \\
 & | & (n).P & \textit{anonymous input} \\
 & | & \langle n \rangle & \textit{anonymous output}
\end{array}
$$

M stands for the *capabilities* defined by the following grammar:

$$
\begin{array}{rcll}
M & | & \mathsf{in}\, n & \textit{enter capability} \\
 & | & \mathsf{out}\, n & \textit{exit capability} \\
 & | & \mathsf{open}\, n & \textit{open capability} \\
 & | & \overline{\mathsf{in}}\, n & \textit{be entered capability} \\
 & | & \overline{\mathsf{out}}\, n & \textit{be exit capability} \\
 & | & \overline{\mathsf{open}}\, n & \textit{be opened capability}
\end{array}
$$

All the processes have the same informal meaning as described in Section 2.2.1 for MA. There are only three new co-capabilities, that express that the ambient synchronises in the computation: the co-capability $\overline{\mathsf{out}}\, n$ expresses that an ambient n is willing to release an internal ambient; the co-capability $\overline{\mathsf{in}}\, n$ expresses that an ambient n is willing to accept another entering ambient and the co-capability $\overline{\mathsf{open}}\, n$ expresses that an ambient n can be opened. The set of free names of P is written $\mathit{fn}(P)$ and is defined in the standard way as in Definition 2.2.2, taking into account the three co-capabilities.

DEFINITION 2.3.2 Structural congruence $\equiv_s$ for SA is identical to Definition 2.2.5.

DEFINITION 2.3.3 The *reduction relation* $\longrightarrow_s \subseteq \mathcal{P}r_s \times \mathcal{P}r_s$ is the smallest relation satisfying the following set of rules:

$$
\begin{array}{rcll}
m\,[\mathsf{in}\, n.P_1 \mid P_2] \mid n\,[\overline{\mathsf{in}}\, n.Q_1 \mid Q_2] & \longrightarrow_s & m\,[P_1 \mid P_2 \mid n\,[Q_1 \mid Q_2]] & \text{RED IN} \\
n\,[\overline{\mathsf{out}}\, n.P_1 \mid P_2 \mid m\,[\mathsf{out}\, n.Q_1 \mid Q_2]] & \longrightarrow_s & n\,[P_2 \mid P_2] \mid m\,[Q_1 \mid Q_2] & \text{RED OUT} \\
\mathsf{open}\, n.P \mid n\,[\overline{\mathsf{open}}\, n.Q_1 \mid Q_2] & \longrightarrow_s & P \mid Q_1 \mid Q_2 & \text{RED OPEN} \\
\langle q \rangle \mid (n).P & \longrightarrow_s & P\{q/n\} & \text{RED COMM}
\end{array}
$$

$$
\frac{P \longrightarrow_s P'}{P \mid R \longrightarrow_s P' \mid R}\ \text{RED PAR} \qquad \frac{P \longrightarrow_s P'}{(\nu n)P \longrightarrow_s (\nu n)P'}\ \text{RED RESTR}
$$

$$\frac{P \longrightarrow_s P'}{n\,[P] \longrightarrow_s n\,[P']}\ \text{RED AMB} \qquad \frac{P \equiv_s P' \longrightarrow_s Q' \equiv_s Q}{P \longrightarrow_s Q}\ \text{RED CONG}$$

DEFINITION 2.3.4 (*Strong barbs*) A process P *exhibits a barb* n, written as $P \downarrow_{\overline{open}} n$, if and only if, $P \equiv_a (\nu p_1 \ldots p_n)(n\,[\overline{\mathsf{open}}\,n.P' \mid P''] \mid P''')$ for some P', P'', P'' and $n \notin \{p_1 \ldots p_n\}$.

REMARK 2.3.5 The choice of *barb* here differs from the one in the literature [48]. In [48] the name of an ambient is observable if it contains also the co-capability that allows ambients to enter. The previous barb is sufficient for the purpose of this thesis. Furthermore, our definition does not make any difference in the definition of contextual barbed bisimilarity,as proved in [87, 52].

The notion of weak barb $P \Downarrow_{\overline{open}} n$ is derivable as in Definition 1.2.2. Hence, by means of the framework described in Chapter 1 we formally define SA as:

$$\mathsf{SA} \stackrel{def}{=} \langle \mathcal{P}r_s,\ \longrightarrow_s, \downarrow_{\overline{open}} \rangle$$

. We write SA^* $\stackrel{def}{=}$ $\langle \mathcal{P}r_s,\ \longrightarrow_s, \downarrow_{\overline{open}} \rangle$ for the pure version without communication primitives. Finally we can present the definition of the equivalences as in the spirit of Chapter 1.

DEFINITION 2.3.6

- We write $\dot{\approx}_s$ for barbed bisimilarity defined over SA as in Definition 1.3.4.
- We write $\cong^c_s$ for barbed congruence defined over SA as in Definition 1.3.7.
- We write $\approx_s$ for contextual barbed bisimilarity defined over SA as in Definition 1.3.11.

Roughly speaking, SA can be viewed as least as expressive as MA since there exists an obvious encoding of MA into SA. We report below the clause for the ambient. The definition for the other operator is homomorphic.

$$[\![n\,[P]]\!] \stackrel{def}{=} n\,[!\overline{\mathsf{out}}\,n \mid !\overline{\mathsf{in}}\,n \mid \overline{\mathsf{open}}\,n \mid [\![P]\!]].$$

Chapter 3

Encoding of the matching operator

In this chapter we shall present the encoding of the *matching operator* in MA. The matching operator was introduced in the π-calculus [61], and under certain conditions it cannot be encoded into some dialect of the π-calculus. First, we shall discuss what can be considered a 'good' encoding. Then follows the impossibility result of the encoding of matching for the π-calculus with mixed choice due to Carbone and Maffeis [17], and the impossibility result of the encoding of matching for the asynchronous π-calculus, which is derived by using full-abstraction properties of contextual equivalence on special processes called *equators* [46]. Finally, we shall present the encoding of the matching operator in MA and we will show that this encoding satisfies at least the same specific conditions under which previous encodings in dialects of the π-calculus failed. Quite interestingly, the work on the encoding of the matching operator gives an insight for comparison between the π-calculus and MA. In fact, we establish that the pure version of MA cannot be encoded into the π-calculus with mixed choice but without the matching operator. We conclude the chapter with an extended example of a router for sending e-mails. We claim that in this example the matching operator is extremely useful.

3.1 *What is an encoding?*

In this section, we shall review the notion of encoding, and aim to describe the strength of some of the properties that an encoding is expected to satisfy.

An *encoding* is a function between two languages, which preserves 'relevant' semantic prop-

erties. For example, consider two calculi

$$\mathsf{C} \stackrel{def}{=} \langle S,\ \longrightarrow, \downarrow \rangle \qquad\qquad \mathsf{C}^* \stackrel{def}{=} \langle T,\ \longrightarrow^*, \downarrow^* \rangle.$$

Moreover assume the existence of a function $[\![\ ]\!] : S \to T$. S is called the *source language* while T is called the *target language*[1]. In general, if an encoding exists, we refer to this as *a positive result,* as opposed to a *negative result* when encodings do not exist.

First of all, we are *not* keen to accept any function $[\![\ ]\!]$ from source language to target language as an encoding, unless some 'relevant' behaviour of the first language is 'preserved'.

Moreover, we appeal here to the intuitive meaning of the words 'relevant' and 'to preserve', but it remains to formalize the meaning of these words, by exhibiting a list of (semantic) properties that $[\![\ ]\!]$ must satisfy. It would avoid disputes, if the scientific community had agreed on a list of possible properties that are regarded as *necessary,* for a function to be upgraded to the rank of 'encoding'. Unfortunately, there is no such definitive list; however we shall give below some of the most common properties. Let $\asymp$ stand for any possible semantic equivalence on S, and $\asymp^*$ on T.

*preservation of execution steps (*completeness*)* : if $P \longrightarrow P'$ then $[\![P]\!] \longrightarrow\!\!\!\!\!\rightarrow\ ^*[\![P']\!]$ [67, 56, 23, 73].

*reflection of execution steps (*soundness*)* : if $[\![P]\!] \longrightarrow^* P'$ then for some R, $P \longrightarrow\!\!\!\!\!\rightarrow\ R$ and $[\![R]\!] \asymp^*\ P'$ [67, 56, 73].

semantic equivalence : if $[\![\ ]\!] : T \longrightarrow T$ then $P \asymp^*\ [\![P]\!]$ [67].

*barb reflection (*soundness*)* : if $[\![P]\!] \Downarrow^*\ n$ then $P \Downarrow\ n$ [73, 90] .

*barb preservation (*completeness*)* : if $P \Downarrow n$ then $[\![P]\!] \Downarrow^*\ n$ [73, 90].

adequacy of the encoding : if $[\![P]\!] \asymp^*\ [\![Q]\!]$ then $P \asymp\ Q$ [56, 76, 67, 73, 90].

completeness of the encoding : if $P \asymp Q$ then $[\![P]\!] \asymp^*\ [\![Q]\!]$ [76, 67, 90].

(Of course, other properties could be added).

The first two conditions together (soundness and completeness of preservation of execution steps) are known as *operational correspondence.* If an encoding satisfies the conditions of barb preservation and barb reflection then we say that the encoding is *computationally adequate.* The combination of the last two conditions (adequacy and completeness of the encoding) is known as *full abstraction.*

[1] Although we consider here calculi with reduction semantics, the following considerations would be equally true for other semantics.

One might also add syntactic requirements on an encoding. To give a concrete example, assuming that $|$ and ν are two operators common to S and T: then the statements below express that $[\![\ ]\!]$ preserves bound names (restriction) and distribution (composition). Clearly the list could be longer, according to the number of common operators in the source and the target language. Other syntactic properties specific to languages could be considered.

distribution preservation : $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$ [71, 74].

name preservation : $[\![(\nu n)P]\!] = (\nu n)[\![P]\!]$ [28].

substitution preservation : for all substitutions σ on S there exists a substitution θ on T such that $[\![\sigma(P)]\!] = \theta([\![P]\!])$;

link independence : if $\mathit{fn}(P) \cap \mathit{fn}(Q) = \emptyset$ then $\mathit{fn}([\![P]\!]) \cap \mathit{fn}([\![Q]\!]) = \emptyset$ [71].

(Of course, other properties could be added.)

The list of properties given above is certainly not exhaustive, but it includes those properties (and variations of them) that are going to be discussed later in this chapter. Moreover, it contains the most common properties used by the scientific community [24, 55, 58, 67, 65, 75, 71, 90, 91].

In general, it is not required that all of the properties above are satisfied in order for a function to be called an encoding (or a 'good' encoding). More specifically, there is not even a subset of these properties that is regarded as *necessary*. For instance, Cardelli and Gordon [24] establish a weak form of preservation of reduction steps (completeness) for the encoding of the asynchronous π-calculus into MA. On the other hand, the encoding of the lazy λ-calculus into the π-calculus [56] proves some variant of both properties for the preservation of execution steps and adequacy. The encoding of the polyadic π-calculus into the monadic one with graph types was proved fully abstract by [90, 91]. The encoding of input guarded choice in the asynchronous π-calculus [67] respect all the operators (apart from choice, of course), and it is fully abstract with respects to different equivalences (weak bisimulation and couple bisimulation). On the other hand, for the encoding of the mixed choice in [65] Nestmann explicitly claimed that distribution preservation is a too strong requirement for an encoding. Prasad [75] shares the same opinion. At the exact opposite end, Palamidessi [71] imposes distribution preservation for the negative result on mixed choice. On this particular point there will be a more detailed discussion in Chapter 6. It must be observed that the controversial examples of encodings given above must not be interpreted as a failure of theoretical computer science, but are dictated by the enormous spectrum of purposes that encodings serve. Thus criteria for encodings are not arbitrary, but aim to guarantee that the purpose of the encoding is fulfilled.

Fortunately, the situation is not *too* wild. It is 'agreed' that some properties are weaker (or stronger) than others. Given the function $[\![\ ]\!]$, the stronger are the properties preserved, the more

inclined we are to accept $[\![\ ]\!]$ as an encoding (or as a good encoding, as Nestmann discusses [65]). Ultimately, the problem of defining what a (good) encoding is, concerns the strength of the properties that $[\![\ ]\!]$ preserve,s as Nestmann [65] clearly states. It is commonly accepted that adequacy of the encoding and completeness of the encoding are *very strong* properties. In particular (Nestmann as rightly points out), adequacy might be easier to establish as opposed to completeness, which has to take care of extra steps introduced by the encodings. In many cases, completeness does not hold because in the case that $\asymp$ is a congruence, contexts that are not in the image of $[\![\ ]\!]$ might be problematic. One example is the encoding of the polyadic π-calculus [57] into the monadic one. This matter is discussed also in Section 4.4 for the encoding of the asynchronous π-calculus into the Push and Pull Ambient Calculus. In the case that preservation of semantic equivalence is impossible to establish, one might regard operational correspondence as preservation or reflection of execution steps as a good alternative, which could establish whether there is an equivalence between terms of the source language and encoded terms, in the case that the encoding is dealing with the same source and target language.

However, in some cases, also these properties (or variation of them) could be too strong, and one might resort to *barb preservation,* or other ad hoc properties. In addition to semantic properties one might require syntactic properties to be preserved. For instance, in encodings of languages that admit a solution for leader election problems, usually it is required that the encoding is homomorphic with respect to composition, i.e. preserves distribution. This requirement aims to avoid that the encoding introduces a trivial solution to such a problem [71]. However, also this condition can be regarded as too strong as Nestmann and Prasad argue [67, 75].

Although there is no unanimous agreement on what an encoding constitutes, or what a ‘good ’encoding is, it is clear that the judgment whether a function is an encoding relies on acceptance or rejection of the properties that hold for the encoding. Some authors adopt a subset of the properties listed above as a guideline for a (possibly good) encoding [67, 65, 91, 71, 24, 56, 76]. However, the scientific community would agree that it is too strong a requirement to ask that an encoding satisfies all the properties listed above. The encoding of the matching that we shall give in Section 3.3 satisfies all those conditions. In this precise sense, we claim that the encoding is *very robust*. In fact, not only the semantic properties of full abstraction, operational correspondence and preservation of barbs hold, but also the syntactic properties on preservation of substitution and all operators except for matching.

The concept of encoding is inherently associated to expressiveness. Namely, if there exists an encoding from the source language S to the target language T, one could interpret this result as the fact that the calculus C^* ‘mirrors’ the calculus C. Thus we could say that in some sense C^* is at least as expressive as C. Expressiveness results, hence encodings, are sought for different

reasons. For instance one could show that some primitives are redundant in a calculus by showing an encoding from the full set of processes to an appropriate fragment. This could be very useful for implementation purposes. This is the case of the programming language PICT, which is based on the asynchronous π-calculus where input-guarded choice can be implemented [67]. One could also show that one calculus can be encoded into another in order to 'inherit' some (possibly good) properties. For instance, from the encoding of the λ-calculus into the π-calculus one could derive easily the Turing completeness of the π-calculus. Since the encoding of the matching operator $[n = m]P$ as presented here is very robust in the sense specified above, we contend that there is no *gain of expressiveness* in adding matching in MA. On the contrary, this operator is *faithfully encodable* in the sense that it is semantically impossible to distinguish the calculus with or without matching.

Before introducing the encoding of the matching operator and proving its properties, we are going to show that under two different subsets of conditions written above, the matching operator cannot be encoded into the π-calculus (with mixed choice); on the other hand, in MA, matching satisfies exactly these conditions.

3.2 *Non-encodability of matching in the π-calculus*

In this section we shall show in two different ways, that the matching operator cannot be encoded into dialects of the π-calculus. Each of the results in the following section appeals to a subset of properties described in the previous section. The first result we present was established by Carbone and Maffeis [17], for the π-calculus with mixed choice. Below, we introduce some additional material, that will be useful.

DEFINITION 3.2.1 Let $P \in \mathcal{P}r_\pi$. $P \Downarrow$ if and only if there exists a n such that $P \Downarrow n$.

LEMMA 3.2.2 Let $P \in \mathcal{P}r_\pi$ and and a let σ any substitution. Assume $P \not\Downarrow$. If $\sigma(P) \longrightarrow\!\!\!\!\!\rightarrow_\pi \sigma(P')$ then $P \longrightarrow\!\!\!\!\!\rightarrow_\pi P'$. Moreover for all substitutions σ we have $\sigma(P) \not\Downarrow$.

Proof. Assume that $P \not\Downarrow$. By induction on $\longrightarrow_\pi$ one can prove that if $\sigma(P) \longrightarrow_\pi \sigma(P')$ then $P \longrightarrow_\pi P'$. Then with an induction on the reduction steps, of $\longrightarrow\!\!\!\!\!\rightarrow_\pi$ the lemma is proved.

The conditions for the encodings are given in the following definition.

DEFINITION 3.2.3 Let L, L^* be process languages. An encoding $[\![-]\!] : L \to L^*$ is

i) *substitution-preserving* if for any substitution of names σ in L there exists a substitution θ in L^* such that $[\![\sigma(P)]\!] = \theta([\![P]\!])$.

ii) *weak barb-respecting* if for any P in L, $P \Downarrow$ iff $[\![P]\!] \Downarrow$.

THEOREM 3.2.4 There does not exist a substitution-preserving and weak barb-respecting encoding from $\pi_m^=$ into π_m.

Proof. Assume there exists such an encoding $[\![-]\!]$. Let $m \neq m$ and σ be a substitution with $\sigma(n) = m$ and all other names unchanged. Let $P = [n = m]\overline{n}\langle r\rangle$. We have $\sigma(P) \Downarrow$ but $P \not\Downarrow$. By Definition 3.2.3(*i*) there is a substitution θ satisfying $[\![\sigma(P)]\!] = \theta([\![P]\!])$. But since $\sigma(P) \Downarrow$ we have $[\![\sigma(P)]\!] \Downarrow$ (weak barb-respecting condition). So $\theta([\![P]\!]) \Downarrow$. By Lemma 3.2.2 we have $[\![P]\!] \Downarrow$. Hence $P \Downarrow$, which is a contradiction.

In order to accept this result, one has to agree that it is 'reasonable' or 'acceptable' for an encoding to be substitution-preserving and weak barb-respecting. Substitutions modify the semantics of calculi; for this reason substitution-preserving is considered quite a strong requirement. For instance consider the following CCS process:

$$a.b \mid \overline{c}.b \not\rightarrow$$

Apply an non-injective substitution σ such that $\sigma(a) = c$ and it behaves like the identity on the other names. Then, the number of possible reductions increases:

$$\sigma(a.b \mid \overline{c}.b) \longrightarrow b \mid b.$$

In the π-calculus, the input prefix acts as a form of non-injective substitution. Consider the process $P \stackrel{def}{=} a(x).(\overline{x}\langle x\rangle.\overline{b}\langle h\rangle \mid c(v).\overline{b}\langle h\rangle)$. The process P is similar in spirit to the previous CCS since if P is composed with $Q \stackrel{def}{=} \overline{a}\langle c\rangle$ then $P \mid Q \Downarrow \overline{b}$. Otherwise it will be deadlocked. In the π-calculus the substitution-preserving condition, as in Definition 3.2.3, is *acceptable* on the basis that the semantics of the π-calculus has to take into account non-injective substitutions.

The next result deals with the asynchronous π-calculus. It is based on well-known facts; thus we do not claim any originality in Proposition 3.2.5 and Proposition 3.2.7 which were proved in [46]. Nevertheless, it remains quite interesting to see that the same conditions that lead to a negative result for the encodability of the matching operator for the asynchronous π-calculus, hold for the positive result on MA. First of all, we recall that for the asynchronous π-calculus, there are some interesting processes [46]. We recall the ones that are going to be useful here:

$$\mathcal{E}(b, c) \stackrel{def}{=} !c(x).\overline{b}\langle x\rangle \mid !b(x).\overline{c}\langle x\rangle$$

The process $\mathcal{E}(b, c)$ is called an *equator*. This is a process that equates processes that emit an output on channel b and on channel c. Moreover, in the asynchronous π-calculus without matching π_a,

this equator fails to distinguish the message of the output, as the following well-known proposition shows.

PROPOSITION 3.2.5 Let $\mathcal{E}(c,b), \mathcal{E}(b,c) \in \mathcal{P}r_{\pi_a}$. Then

$$\mathcal{E}(c,b) \mid \overline{a}\langle b\rangle \approx_{\pi_a} \mathcal{E}(b,c) \mid \overline{a}\langle c\rangle.$$

Using contextual bisimulation, the two processes are indistinguishable. Intuitively, the only context that could interact with those processes is the one that either inputs something on channel a or outputs something on channel b or c. It must be observed that the two processes differ on the name being output on the channel a. To make this visible as a barb, we should build a context such as $a(x).\overline{x}\langle m\rangle$, but the equator does not distinguish c from b, thus the equivalence. A formal proof of this can be found in [46]. The previous fact tells us that contextual barbed bisimulation is not able to distinguish the names that are output by the channel a.

Assume that we are given two languages $L^{=}$ and L, such that the syntax of $L^{=}$ is obtained by augmenting L with the matching operator $[n=m]P$ only. One possible encoding $[\![-]\!] : L^{=} \longrightarrow L$ might preserve all the operators apart from the matching. The next definition formalizes a notion of hyper-robustness, a fully abstract encoding that is the identity on all operators different from matching. We do not propose this a suitable notion of encoding, as explained earlier; it is an ad hoc definition to prove that the asynchronous π-calculus with matching is not encodable in the asynchronous π-calculus. In the definition below, we write $\approx$ and $\approx^{=}$ for contextual barbed bisimilarities as defined in Section 1.3.3.

DEFINITION 3.2.6 Let $L^{=}, L$ be process languages, such that the grammar of $L^{=}$ is obtained by adding the matching operator to the grammar of L. An encoding of $[\![-]\!] : L^{=} \to L$ is *hyper-robust* if:

i) it almost preserves identity: if $P \in L$ then $[\![P]\!] = P$;

ii) it is fully abstract: for all $P, Q \in L^{=}$, $[\![P]\!] \approx [\![Q]\!]$ if and only if $P \approx^{=} Q$.

THEOREM 3.2.7 There does not exist a hyper-robust encoding from $\pi_a^{=}$ into π_a.

Proof. Assume that there exists a hyper-robust encoding $[\![-]\!]$, then

$$\begin{aligned} [\![\mathcal{E}(b,c) \mid \overline{a}\langle c\rangle]\!] &= \mathcal{E}(b,c) \mid \overline{a}\langle c\rangle \text{ by } (i) \text{ of Def. 3.2.6} \\ [\![\mathcal{E}(c,b) \mid \overline{a}\langle b\rangle]\!] &= \mathcal{E}(c,b) \mid \overline{a}\langle c\rangle \text{ by } (i) \text{ of Def. 3.2.6} \end{aligned}$$

By Proposition 3.2.5, $\mathcal{E}(c,b) \mid \overline{a}\langle b\rangle \approx_{\pi_a} \mathcal{E}(b,c) \mid \overline{a}\langle c\rangle$ and by Definition 3.2.6(*ii*) we conclude

$$\mathcal{E}(c,b) \mid \overline{a}\langle b\rangle \approx_{\pi_a^{=}} \mathcal{E}(b,c) \mid \overline{a}\langle c\rangle \tag{3.1}$$

Then, consider the context $\mathcal{C} = a(x)[x=b]\overline{\omega}\langle t\rangle \mid [-]$ generated by the syntax $\mathcal{P}r_{\pi_a^=}$.

$$\mathcal{C}[\mathcal{E}(c,b) \mid \overline{a}\langle b\rangle] \Downarrow \omega \text{ and } \mathcal{C}[\mathcal{E}(c,b) \mid \overline{a}\langle c\rangle] \not\Downarrow \omega.$$

Thus, we conclude that

$$\mathcal{E}(c,b) \mid \overline{a}\langle b\rangle \not\approx_{\pi_a^=} \mathcal{E}(b,c) \mid \overline{a}\langle c\rangle$$

which is a contradiction with the statement in equation (3.1). Thus a hyper-robust encoding from $\pi_a^=$ to π_a does not exist.

The previous result is based upon very strong conditions for the encoding, which make a negative result less significant. In fact, we certainly have proved that a hyper-robust encoding does not exist, but we cannot exclude the existence of a weaker encoding, for instance an encoding that is barb-respecting only. It must be noted that a barb-respecting and substitution-preserving encoding of matching does not exist in the asynchronous π-calculus. In fact, Proposition 3.2.2 would be valid in the asynchronous π-calculus setting as well, thus Theorem 3.2.4 would be proved in the same way. Moreover, the two notions of encoding are not comparable in general, in the sense that they do not seem to imply each other. It is obvious that barb-respecting and substitution-preserving does not imply (in general) hyper-robustness. The other way around is also true, since hyper-robustness certainly implies the barb-respecting condition; however it does not imply substitution-preservation.

3.3 *Encoding of the matching*

In this section we present the encoding of the matching operator in MA which was given for the first time in the dialect of MA, the Push and Pull Ambient Calculus [73], and also in [74]. We will see that this encoding is very simple and natural; moreover it is hyper-robust and substitution-preserving. If the matching operator were a primitive operator in MA, it would yield a different language $\mathcal{P}r_{a^=}$. The reduction semantics would change by adding the following rule to structural equivalence:

$$[n=n]P \quad \equiv_a^= \quad P. \tag{3.2}$$

We recall that the matching $[n=w]P$ behaves like P if and only if $n = w$.

By the means of the framework described in Chapter 1 we formally define MA with matching as:

$$\mathsf{MA}^= \stackrel{def}{=} \langle \mathcal{P}r_{a^=}, \longrightarrow_a, \downarrow \rangle.$$

DEFINITION 3.3.1 The encoding of the matching is the function $[\![\ ]\!] : \mathcal{P}r_{a=} \longrightarrow \mathcal{P}r_a$ defined as follows:

$$
\begin{array}{rcl}
[\![\mathbf{0}]\!] & \stackrel{def}{=} & \mathbf{0} \\
[\![M.P]\!] & \stackrel{def}{=} & M.[\![P]\!] \\
[\![[m=n]P]\!] & \stackrel{def}{=} & (\nu z\, y)(z\,[\mathsf{open}\, m.y\,[\mathsf{out}\, z] \mid n\,[\,]] \mid \mathsf{open}\, y.\mathsf{open}\, z.[\![P]\!]) \\
 & & \qquad\qquad \text{with } y, z \notin \mathit{fn}([\![P]\!]) \\
[\![P \mid Q]\!] & \stackrel{def}{=} & [\![P]\!] \mid [\![Q]\!] \\
[\![(\nu n)P]\!] & \stackrel{def}{=} & (\nu n)[\![P]\!] \\
[\![n\,[P]]\!] & \stackrel{def}{=} & n\,[[\![P]\!]] \\
[\![(w).P]\!] & \stackrel{def}{=} & (w).[\![P]\!] \\
[\![\langle n \rangle]\!] & \stackrel{def}{=} & \langle n \rangle \\
[\![!P]\!] & \stackrel{def}{=} & !\,[\![P]\!]
\end{array}
$$

The encoding is fully homomorphic, in the sense that it preserves all the common operators. Moreover, the protocol that implements the matching is fully deterministic, in the sense that there exists only one possible path of reduction. Moreover, there is no garbage left, in the sense that, if a term P is encoded, after the reduction of $[\![P]\!]$, the residual process left is the encoding of the residual of the original term, and nothing more. Besides, since communication primitives are fully disposable, the encoding would work for the MA without communication primitives.

Observing that $\mathit{fn}([n=m]P) = \mathit{fn}(P) \cup \{n, m\}$, the following propositions hold.

LEMMA 3.3.2 *i)* $\mathit{fn}(P) = \mathit{fn}([\![P]\!])$.

ii) If $P \in \mathcal{P}r_a$ then $[\![P]\!] = P$.

iii) For any substitution σ, $\sigma([\![P]\!]) = [\![\sigma(P)]\!]$.

Proof. For (*i*) and (*ii*), the proof is immediate by following the clauses in Definition 3.3.1. Part (*iii*) follows from part (*i*) and Definition 3.3.1.

PROPOSITION 3.3.3 *i)* $[\![[n=n]P]\!] \longrightarrow\!\!\!\!\!\rightarrow_a [\![P]\!]$.

ii) If $n \neq m$ then $[\![[n=m]P]\!] \not\longrightarrow_a$.

iii) $[\![P]\!] \lesssim [\![[n=n]P]\!]$.

Proof. *i)* Immediate from the reduction steps of Definition 3.3.1.

ii) Immediate from the reduction steps of Definition 3.3.1.

iii) For readability let us define:

$$\begin{aligned} P' &\stackrel{def}{=} (\nu z\, y)(z\,[y\,[\mathsf{out}\, z]] \mid \mathsf{open}\, y.\mathsf{open}\, z.[\![P]\!]) \\ P'' &\stackrel{def}{=} (\nu z\, y)(z\,[\,] \mid y\,[\,] \mid \mathsf{open}\, y.\mathsf{open}\, z.[\![P]\!]) \\ P''' &\stackrel{def}{=} (\nu z\, y)(z\,[\,] \mid \mathsf{open}\, z.[\![P]\!]) \end{aligned}$$

Consider the following relations $\mathcal{R}_i \subseteq \mathcal{P}r_a \times \mathcal{P}r_a$:

$$\begin{aligned} \mathcal{R}_1 &\stackrel{def}{=} \{(\mathcal{C}[[\![P]\!]], \mathcal{C}[P'''])\} \cup \mathsf{id} \\ \mathcal{R}_2 &\stackrel{def}{=} \{(\mathcal{C}[P'''], \mathcal{C}[P''])\} \cup \mathsf{id} \\ \mathcal{R}_3 &\stackrel{def}{=} \{(\mathcal{C}[P''], \mathcal{C}[P'])\} \cup \mathsf{id} \\ \mathcal{R}_4 &\stackrel{def}{=} \{(\mathcal{C}[P'], \mathcal{C}[[\![[n=n]P]\!]])\} \cup \mathsf{id} \end{aligned}$$

A long induction on contexts is necessary to verify that for each $1 \leq i \leq 4$ $\mathcal{R}_i$ is an a weak contextual simulation and $\mathcal{R}_i^{-1}$ is a strict simulation: thus $\mathcal{R}_i \subseteq \lesssim$. By transitivity of $\lesssim$ and Proposition 1.3.17(*ii*), the result follows.

REMARK 3.3.4 Usually, the hurdle in a proof similar to the above is to deal with contexts that act as non-injective substitutions, for example $(n).[\,] \mid \langle m \rangle$, since they might trigger the computation of a deadlocked process that has different names in the matching operator, such as $[n = m]P$. However, it must be observed that, by assumption all the names in the matching operator are equal $[q = q]P$, and so such contexts do not interfere with the protocol that simulates the matching operator.

REMARK 3.3.5 We write $\equiv_a^=$ for the structural congruence relation augmented with equation 3.2. It is immediate to verify that $\equiv_a \subset \equiv_a^=$. This observation will be useful in the next proposition.

PROPOSITION 3.3.6 *i*) Let $P, Q \in \mathcal{P}r_a$. If $P \equiv_a Q$ then $[\![P]\!] \equiv_a [\![Q]\!]$.

ii) Let $P, Q \in \mathcal{P}r_{a^=}$. If $P \equiv_a^= Q$ then $[\![P]\!] \approx_a [\![Q]\!]$.

Proof. The first part of the proposition is trivial by Lemma 3.3.2(*ii*). The second part is derivable from the first by observing that $\equiv_a \subset \approx_a$. The case for matching follows from Proposition 3.3.3(*iii*) and from the fact that the symmetric closure of $\lesssim$ is included in $\approx_a$.

One could wonder why, in Proposition 3.3.6(*i*), $\approx_a$ is present for the translation of the processes but not $\lesssim$, since $[\![[n=n]]\!]P \lesssim [\![P]\!]$. In order to give an intuition, we invite the reader to consider the following process: $[n = n]r\,[P] \equiv_a^= r\,[[s=s]P]$. Now, since $[\![[s=s]P]\!] \lesssim [\![P]\!]$ and $[\![[n=n]r\,[P]]\!] \lesssim [\![r\,[P]]\!]$, the symmetric closure of $\lesssim$ is $\approx_a$, thus the result in Proposition 3.3.6(*i*).

The next proposition tells us that barbs are respected by the encoding.

PROPOSITION 3.3.7 Let $P, Q \in \mathcal{P}r_{a^=}$, then:

i) For all names n, if $P \downarrow n$ then $[\![P]\!] \Downarrow n$.

ii) For all names n, if $[\![P]\!] \downarrow n$ then $P \downarrow n$.

Proof. *i*) Assume $P \downarrow n$, then either

$$\begin{array}{lll} P & \equiv_a^= & (\nu p_1 \ldots p_n)(n\,[P'] \mid P'') \qquad \text{or} \\ P & \equiv_a^= & [r=r](\nu p_1 \ldots p_n)(n\,[P'] \mid P''), \end{array}$$

for some P', P'' and $n \notin \{p_1 \ldots p_n\}$. In the first case, by Definition 3.3.1 we conclude $[\![P]\!] \downarrow n$. In the second case by Proposition 3.3.6(*i*).

ii) The proof is immediate by Definition 3.3.1 and definition of barbs.

In the light of the previous results on the structural congruences, we can show the first correspondence proposition.

PROPOSITION 3.3.8 Let $P \in \mathcal{P}r_{a^=}$. If $P \longrightarrow_{a^=} P'$, then for some R, S
$[\![P]\!] \approx_a S \longrightarrow\!\!\!\!\rightarrow_a R \approx_a [\![P']\!]$.

Proof. The proof goes by induction on the derivation of $\longrightarrow_{a^=}$.
RED IN : Assume $m\,[\mathsf{in}\, n.P \mid Q] \mid n\,[R] \longrightarrow_{a^=} n\,[m\,[P \mid Q] \mid R]$. Then:

$$\begin{array}{rcll} [\![m\,[\mathsf{in}\, n.P \mid Q] \mid n\,[R]]\!] & = & m\,[\mathsf{in}\, n.[\![P]\!] \mid [\![Q]\!]] \mid n\,[[\![R]\!]] & \text{by Def. 3.3.1} \\ & \longrightarrow_a & n\,[m\,[[\![P]\!] \mid [\![Q]\!]] \mid [\![R]\!]] & \text{by RED IN} \\ & = & [\![n\,[m\,[P \mid Q] \mid R]]\!] & \text{by Def. 3.3.1} \end{array}$$

Observing that $\mathsf{id} \subset \approx_a$, the proposition is proved. For the rules RED OUT, RED OPEN, and RED COMM, the proof is very similar to the previous case. For the rules RED PAR, RED RESTR, and RED AMB the proof relies on induction hypothesis and transitivity of $\approx_a$.

The interesting case is the rule RED CONG. Assume $P \equiv_a^= P' \longrightarrow_{a^=} Q' \equiv_a^= Q$. Then by induction there exist S, R such that $S \longrightarrow\!\!\!\!\rightarrow_a R$ and $S \approx_a [\![P']\!]$ and $R \approx_a [\![Q']\!]$. By Proposition 3.3.6(*i*) $[\![Q']\!] \approx_a [\![Q]\!]$ and $[\![P]\!] \approx_a [\![P']\!]$. Thus by transitivity of $\approx_a$ we conclude

$$[\![P]\!] \approx_a [\![P']\!] \longrightarrow\!\!\!\!\rightarrow_a [\![Q']\!] \approx_a [\![Q]\!].$$

The previous result is not useful for proving full abstraction theorem, since it does not allow us to derive a stronger statement on the transitive closure of $\longrightarrow_a$. The real hurdle is the presence of the equivalence $\approx_a$. The equivalence $\approx_a$ is necessary because of the symmetry of the congruence rule $[n=n]P \equiv_a^= P$. For the implementation of matching, it makes sense to use this rule from left to right. From right to left, the rule has no computational meaning. In order to express this fact we are going to use a reduction rule that gives a computational meaning to the matching operator. This rule was used in [33].

$$\frac{P \longrightarrow_a P'}{[n=n]P \longrightarrow_a P'} \text{ RED MATCH}$$

Now, we consider MA with this extra rule for reduction and structural congruence $\equiv_a$. We write $\longrightarrow'_{a=}$ for this slightly different reduction relation. Thus we define $\mathsf{MA}^{='} \stackrel{def}{=} \langle \mathcal{P}r_{a=}, \longrightarrow'_{a=}, \downarrow \rangle$. We define $\approx_a^{='}$ as the contextual barbed bisimulation over $\mathsf{MA}^{='}$.

First we are going to show that the new reduction relation adds no new computational meaning, i.e.the reduction relations coincide.

PROPOSITION 3.3.9 Let $P \in \mathcal{P}r_{a=}$. Then:

i) If $P \longrightarrow'_{a=} P'$ then $P \longrightarrow_{a=} P'$.

ii) If $P \longrightarrow\!\!\!\!\rightarrow'_{a=} P'$ then $P \longrightarrow\!\!\!\!\rightarrow_{a=} P'$.

iii) If $P \longrightarrow_{a=} P'$ then for some R and R' such that $P \equiv_a^= R$ and $P' \equiv_a^= R'$ we have $R \longrightarrow'_{a=} R'$.

iv) If $P \longrightarrow\!\!\!\!\rightarrow_{a=} P'$ then for some R and R' such that $P \equiv_a^= R$ and $P' \equiv_a^= R'$ we have $R \longrightarrow\!\!\!\!\rightarrow'_{a=} R'$.

Proof. Straightforward from the definitions of $\longrightarrow_{a=}$ and $\longrightarrow'_{a=}$.

Moreover contextual barbed equivalence remains the same.

REMARK 3.3.10 In the next proposition, we assume that a weak contextual bisimulation up-to structural congruence is a weak contextual bisimulation. This fact is straightforward to prove [58].

PROPOSITION 3.3.11 Let $P, Q \in \mathcal{P}r_{a=}$. $P \approx_a^{='} Q$ if and only if $P \approx_a^= Q$.

Proof. **if** : We prove that

$$\mathcal{S} = \{(P,Q) : P \approx_a^= Q\}$$

is a weak contextual bisimulation up-to structural congruence. Clearly, if $P \downarrow n$ then $Q \Downarrow n$. Assume that $\mathcal{C}[P] \longrightarrow'_{a=} P'$, then by Proposition 3.3.9(*i*) $\mathcal{C}[P] \longrightarrow_{a=} P'$. Then for some R, R' such that $P \equiv_a^= R$ and $P \equiv_a^= R$, it is the case that $R \longrightarrow'_{a=} R'$ by Proposition 3.3.9(*iii*).

Since $P \approx_a^= Q \approx_a^= R$, there exists a Q' such that $\mathcal{C}[Q] \longrightarrow\!\!\!\!\!\rightarrow'_{a=} Q'$ and $P' \approx_a^= Q'$. By Proposition 3.3.9(*ii*) $\mathcal{C}[Q] \longrightarrow\!\!\!\!\!\rightarrow_a Q'$ which implies that $P' \equiv_a^= R' \; \mathcal{S} \; Q'$. Thus, $(P', Q') \in \mathcal{S}$. The symmetry follows from the symmetry of $\approx_a^=$.

only if : We prove that

$$\mathcal{S} = \{(P, Q) : P \approx_a^{='} Q\}$$

is a weak contextual bisimulation up-to structural congruence. The proof is similar to the previous part with the help of Proposition 3.3.9.

PROPOSITION 3.3.12 Let $P, P' \in \mathcal{P}r_{a^=}$. Then:

i) If $P \longrightarrow'_{a=} P'$ then $[\![P]\!] \longrightarrow_a [\![P']\!]$;

ii) If $P \longrightarrow\!\!\!\!\!\rightarrow'_{a=} P'$ then $[\![P]\!] \longrightarrow\!\!\!\!\!\rightarrow_a [\![P']\!]$;

iii) If $[\![P]\!] \longrightarrow_a Q$ then for some P', $P \longrightarrow\!\!\!\!\!\rightarrow'_{a=} P'$ and $Q \lesssim [\![P']\!]$;

iv) If $[\![P]\!] \longrightarrow\!\!\!\!\!\rightarrow_a Q$ then for some P', $P \longrightarrow\!\!\!\!\!\rightarrow'_{a=} P'$ and $Q \lesssim [\![P']\!]$;

Proof. *i*) By induction on the derivation of $\longrightarrow'_{a=}$. Most of the work has been done in Theorem 3.3.8 for all the rules except RED CONG. The latter is a consequence of Lemma 3.3.6(*i*) and induction hypothesis.

ii) By induction on the number of steps of $\longrightarrow\!\!\!\!\!\rightarrow'_{a=}$.

iii) The proof is an induction on the derivation of $\longrightarrow_a$. We present here only the base cases and we silently make use of the fact that $\mathsf{id} \subseteq \lesssim$. The other cases follow by induction.

– We will show in detail the proof if in the derivation the rule RED OPEN has been used last. Then for some R, T, n it is the case that $[\![P]\!] = [\![\mathsf{open}\, n.R]\!] \mid [\![n\, [T]]\!]$ and

$$\begin{array}{rcl} [\![\mathsf{open}\, n.R]\!] \mid [\![n\, [T]]\!] & = & \mathsf{open}\, n.[\![R]\!] \mid n\, [[\![T]\!]] \\ & \longrightarrow_a & [\![R]\!] \mid [\![T]\!] \\ & = & [\![R \mid T]\!] \end{array}$$

– If $[\![P]\!] = [\![[n=m]P']\!]$ then:

1) if $n \neq m$ then $[\![P]\!] \not\longrightarrow_a$, then it follows by Proposition 3.3.3(*ii*).

2) Otherwise

$$[\![[n=n]P']\!] \;=\; (\nu yz)(\nu z\, y)(z\, [\mathsf{open}\, n.y\, [\mathsf{out}\, z] \mid n\, [\,]] \mid \mathsf{open}\, y.\mathsf{open}\, z.[\![P']\!])$$

with $y, z \notin fn([\![P']\!])$. Assume that

$$[\![[n=n]P']\!] \;\longrightarrow_a\; (\nu yz)(\nu z\, y)(z\, [y\, [\mathsf{out}\, z]] \mid \mathsf{open}\, y.\mathsf{open}\, z.[\![P']\!])$$

Since we can derive: $[n=n]P' \longrightarrow\!\!\!\!\!\rightarrow'_{a^=}\ [n=n]P'$ and by Proposition 3.3.3(*iii*):

$$(\nu yz)(\nu z\, y)(z\,[y\,[\mathsf{out}\, z]] \mid \mathsf{open}\, y.\mathsf{open}\, z.[\![P']\!]) \lesssim [\![[n=n]P']\!].$$

iv) By induction on the number of steps of $\longrightarrow\!\!\!\!\!\rightarrow_a$.

The following corollary says that there are no contexts in MA with matching that interfere with the protocols of the encoding.

COROLLARY 3.3.13 Consider $P \in \mathcal{P}r_a$ and a context $\mathcal{C}$ from the grammar with matching. If $\mathcal{C}[[\![P]\!]] \longrightarrow\!\!\!\!\!\rightarrow'_{a^=}\ P'$ then $[\![\mathcal{C}[P]]\!] \longrightarrow\!\!\!\!\!\rightarrow_a [\![P']\!]$.

Proof. By induction on contexts on the grammar with matching. The only interesting case is if $\mathcal{C} = [n=m]\mathcal{C}'$. There are two cases to consider:

i) $m \neq n$. Then $[n=m][[\![P]\!]]$ does not reduce, and neither does $(\nu z\, y)(z\,[\mathsf{open}\, m.y\,[\mathsf{out}\, z] \mid n\,[\,]] \mid \mathsf{open}\, y.\mathsf{open}\, z.[[\![P]\!]])$.

ii) $m = n$; this follows from Proposition 3.3.9(*i*).

REMARK 3.3.14 In the next proposition we consider the encoding of matching with range in the set of processes with matching,i.e. $[\![\,]\!] : \mathcal{P}r_{a^=} \longrightarrow \mathcal{P}r_{a^=}$. We write $\lesssim^=$ for the expansion defined on processes with matching.

PROPOSITION 3.3.15 *i*) $[n=m]P \lesssim^= [\![[n=m]P]\!]$.

ii) For all $P \in \mathcal{P}r_{a^=}$, $P \lesssim^= [\![P]\!]$.

Proof. *i*) We shall see that $[n=m]P \lesssim^= [\![[n=m]P]\!]$.

For readability let us define:

$$\begin{aligned} P' &\stackrel{def}{=} (\nu z\, y)(z\,[y\,[\mathsf{out}\, z]] \mid \mathsf{open}\, y.\mathsf{open}\, z.[[\![P]\!]) \\ P'' &\stackrel{def}{=} (\nu z\, y)(z\,[\,] \mid y\,[\,] \mid \mathsf{open}\, y.\mathsf{open}\, z.[[\![P]\!]) \\ P''' &\stackrel{def}{=} (\nu z\, y)(z\,[\,] \mid \mathsf{open}\, z.[[\![P]\!]) \end{aligned}$$

Consider the following relations $\mathcal{R} \subseteq \mathcal{P}r_{a^=} \times \mathcal{P}r_{a^=}$:

$$\begin{aligned} \mathcal{R} \stackrel{def}{=}\ & \{(\mathcal{C}[[n=m]P], \mathcal{C}[P'])\} \cup \\ & \{(\mathcal{C}[[n=m]P], \mathcal{C}[P''])\} \cup \\ & \{(\mathcal{C}[[n=m]P], \mathcal{C}[P'''])\} \cup \\ & \{(\mathcal{C}[[n=m]P], \mathcal{C}[[\![[n=m]P]\!]])\} \cup \{P, [\![P]\!]\} \end{aligned}$$

Clearly $\mathcal{R}$ is an a weak contextual simulation and $\mathcal{R}^{-1}$ is a strict simulation: thus $\mathcal{R} \subseteq \lesssim$

ii) It follows from the previous part.

In Section 3.1 we claimed that encodings are intrinsically related to expressiveness. Proposition 3.3.12 says that the encoding $[\![-]\!]$ preserves the intended behaviour of the matching. However, the matching operator influences the semantic equivalence, as one could observe from the proof of Theorem 3.2.7. Thus full abstraction results are crucial in this setting, because they assert that the encoding preserves the semantic equivalence on both target and source languages. This is in general not the case, since if the encoding is a congruence, contexts that are not in the image $[\![-]\!]$ might be problematic for the proof of full abstraction. The encoding of the matching however is fully abstract, meaning that the matching operator does not introduce extra power in MA, in the sense that it does not modify the original contextual barbed bisimulation.

The following theorem of full abstraction shows that the matching operator can be encoded in standard MA, without any loss of expressiveness.

PROPOSITION 3.3.16 (*Full abstraction*) For all $P, Q \in \mathcal{P}r_{a^=}$,
$[\![P]\!] \approx_a [\![Q]\!]$ if and only if $P \approx_a^= Q$.

Proof. **if** : We prove that:

$$\mathcal{S} \stackrel{def}{=} \{(P, Q) : [\![P]\!] \approx_a [\![Q]\!]\}$$

is a weak contextual bisimulation over $\mathsf{MA}^{='}$.

Assume $P \downarrow n$. By Proposition 3.3.7(*i*) $[\![P]\!] \Downarrow n$ which implies $[\![Q]\!] \Downarrow n$ since $[\![P]\!] \approx_a [\![Q]\!]$. By Proposition 3.3.12(*iv*), we conclude $Q \Downarrow n$. Assume for all $\mathcal{C}$, $\mathcal{C}[P] \longrightarrow'_{a^=} P'$; then by Corollary 3.3.13 we have that $[\![\mathcal{C}[P]]\!] \longrightarrow\!\!\!\!\rightarrow_a [\![P']\!]$. Since $[\![P]\!] \approx_a [\![Q]\!]$ then for some Q' we have $[\![Q]\!] \longrightarrow\!\!\!\!\rightarrow_a Q'$ and $[\![P']\!] \approx_a Q'$. By Proposition 3.3.12(*iv*), we get $\mathcal{C}[Q] \longrightarrow\!\!\!\!\rightarrow'_{a^=} T$ and $[\![T]\!] \lesssim Q'$, for some T. Since $\lesssim \subseteq \approx_a$ then $[\![T]\!] \approx_a Q'$ then by transitivity $[\![P']\!] \approx_a [\![T]\!]$. We conclude $(P', T) \in \mathcal{S}$. Thus, $\mathcal{S} \subseteq \approx_a^{='}$. Then the conclusion follows by Theorem 3.3.11.

only if : We prove that:

$$\mathcal{S} \stackrel{def}{=} \{([\![P]\!], [\![Q]\!]) : P \approx_a^{='} Q\}.$$

is a weak contextual bisimulation over $\mathsf{MA}^=$. Since $\lesssim^= \subseteq \approx_a^=$ and $\approx_a^= = \approx_a^{='}$ by Theorem 3.3.11, we have $P \approx_a^{='} Q$. Thus by Proposition 3.3.15 $[\![P]\!] \approx_a^= P \approx_a^{='} Q \approx_a^= [\![Q]\!]$. Clearly, $\mathcal{S}$ is a weak contextual bisimulation over $\mathsf{MA}^=$.

3.4 *Comparing encodings*

In Section 3.2 we have shown that the matching operator could not be encoded into dialects of the π-calculus Theorem 3.2.4 and Theorem 3.2.7 under specific conditions. However, some encodings exist which satisfy different criteria. Whether the conditions are suitable or trivial, depends in ultimate instance on the purpose of the encoding itself. In this section we showed that the encoding of the matching operator in MA satisfies *exactly the same conditions* that were used for the negative result in the π-calculus.

PROPOSITION 3.4.1 The encoding $[\![-]\!] \subset \mathcal{P}r_{a^=} \times \mathcal{P}r_a$ is substitution-preserving and weak barb-respecting.

Proof. The result follows from Proposition 3.3.2(*iii*) and Proposition 3.3.7(*ii*).

PROPOSITION 3.4.2 The encoding $[\![-]\!] \subset \mathcal{P}r_{a^=} \times \mathcal{P}r_a$ is hyper-robust.

Proof. The result follows from Proposition 3.3.2(*ii*) and Theorem 3.3.16.

3.5 *Negative results*

In this section we shall show that pure MA (denoted by MA*) is more expressive than the π-calculus with mixed choice and without matching (π_m). The result has been inspired by [17].

In fact, a restricted ambient might not show any barb, and even be deadlocked; however, because inside the ambient there might be free names, a suitable substitution could trigger a computation that eventually might show a barb, as the process $R \stackrel{def}{=} (\nu z)(z\,[q\,[\,]\mid \mathsf{open}\,s.s\,[\mathsf{out}\,z]])$ illustrates. Observe that $R \not\Downarrow n$ for all n. If σ is a substitution such that $\sigma(q) = \sigma(s)$ then $\sigma(R) \Downarrow \sigma(s)$. As regards the π-calculus, the process $P \stackrel{def}{=} [q = s]Q$ is subject to the same observation, and on this is based the proof in [17]. It is worth noticing that R is the encoding of the matching operator in MA. The following states that pure MA cannot be encoded in the π-calculus with mixed choice but without matching.

THEOREM 3.5.1 There does not exist a substitution-preserving and weak barb-respecting encoding from MA* into π_m.

Proof. Assume there exists such an encoding $[\![-]\!]$. Let σ be a substitution with $\sigma(n) = m$ and all other names unchanged. Let $P \stackrel{def}{=} (\nu\, r)(r\,[\mathsf{open}\,n.m\,[\mathsf{out}\,r]\mid m\,[\,]])$. We have $\sigma(P) \Downarrow$ but $P \not\Downarrow$.

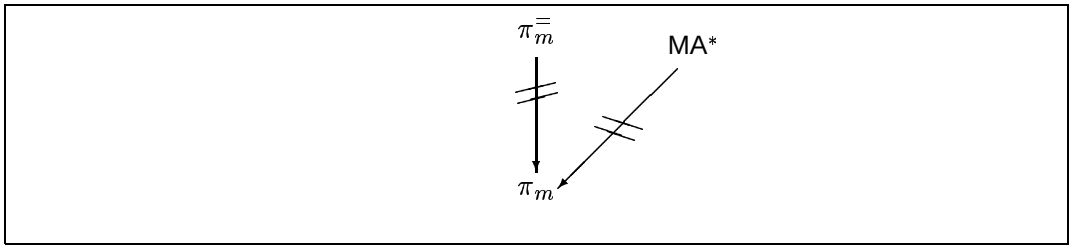
There is a substitution θ satisfying $[\![\sigma(P)]\!] = \theta([\![P]\!])$. But since $\sigma(P) \Downarrow$ we have $[\![\sigma(P)]\!] \Downarrow$ (weak barb-respecting condition). So $\theta([\![P]\!]) \Downarrow$. By Lemma 3.2.2 we have $[\![P]\!] \Downarrow$. Hence $P \Downarrow$, which is a contradiction.

REMARK 3.5.2 The smallest set of primitives necessary to carry out the previous proof would contain only the out capability. In fact, the following process

$$P \stackrel{def}{=} (\nu\, r)(r\,[m\,[s\,[\mathsf{out}\, n.\mathsf{out}\, r]]])$$

could be equally used instead of $P \stackrel{def}{=} (\nu\, r)(r\,[\mathsf{open}\, n.m\,[\mathsf{out}\, r] \mid m\,[\,]])$. Moreover, this proof technique could be adapted to SA quite easily, by considering the following process $n\,[\overline{\mathsf{open}}\, r]$, from which we conclude that the smallest fragment of SA that is not encodable in the π-calculus without matching contains the $\overline{\mathsf{open}}$ capability only.

In Figure 3.1 we show the hierarchy between different calculi. The arrows with two bars on top mean that there is no encoding that is both weak barb-respecting and substitution-preserving.



MA*	=	MA without the communication primitives
$\pi^{=}_{m}$	=	Mixed choice π-calculus with matching
π_m	=	Mixed choice π-calculus

Figure 3.1: Comparing Mobile Ambients against the π-calculus

3.6 *The mail*

We are going to present an extended example of a router delivering mails. The purpose is to show that the matching operator is very useful. The idea is to represent a simplified version of sending mail among users located at different places in the world. The scenario comes from the World Wide Web: at different administrative domains, for example IBM or Imperial College, users send e-mails to each other. The mail is sent to the right place by using a special program called *router*. The router is an ambient that reads the address from the mail and delivers the mail to the addressee.

$$\mathsf{US}\,[\mathsf{MIT} \mid \mathsf{IBM} \mid \mathsf{Stanf} \mid \mathsf{Mcr}] \mid \mathsf{UK}\,[\mathsf{Oxf} \mid \mathsf{IC} \mid \mathsf{Mcr}] \mid \mathsf{Italy}\,[\mathsf{FIAT} \mid \mathsf{IBM} \mid \mathsf{Turin}]$$

Figure 3.2: TOPOLOGY OF LOCATIONS

In [24] a router for packets is defined. In that case, the router knows *one* path for the destination of the *packet*. In order to transfer the path into the packet, there is the need of an output path: thus higher-order communication primitives are needed. For this to work, it is compulsory that the router can deliver to one destination only. In other words, the router cannot choose between two different destinations.

The situation we are trying to represent here is more sophisticated. First of all, there is mail that needs to be delivered. There are two layers of locations: a *virtual location* known by the user and a *real location* known by the router. The user writes the address of the destination. As in real life, the address on the mail is user-transparent, i.e the sender does not know the complete (physical or virtual) path for reaching the destination. The address stands for a *virtual location*, and it gets translated by the router to a *real location*. The router will deliver the mail. We represent the *topology* of real and virtual locations as a composition and nesting of ambients. In Figure 3.2, there is the topology of the virtual location. The domains Mcr and IBM appear in different countries. Thus the router has to distinguish these two different locations of the same administrative domain. Clearly, if there was one domain only, it would be easier to program the router, since for each domain there is one path to follow. We will see also that the user does not need to know that for instance different addresses correspond to the same administrative domain. In our example, the router uses matching to distinguish which one is the suitable path in order to reach the destination.

We need to represent in MA the following elements:

i) the user and the mail;

ii) the administrative domain and the router;

iii) the topology of the network.

We represent here a very simple topology, to keep the example to a manageable size. However the example could in principle be more complicated and more structured. First of all there is the division into countries:

$$\mathsf{US}\,[\ldots] \mid \mathsf{UK}\,[\ldots] \mid \mathsf{Italy}\,[\ldots]$$

In each country there are some administrative domains. These could be intuitively companies

or universities. Each administrative domain contains users, who can send and receive e-mails among each other. We are going to describe these administrative domains inside each country:

$$\begin{aligned} \mathsf{US} &= \mathsf{US}\,[\mathsf{MIT} \mid \mathsf{IBM} \mid \mathsf{Stanf} \mid \mathsf{Mcr}] \\ \mathsf{UK} &= \mathsf{UK}\,[\mathsf{Oxf} \mid \mathsf{IC} \mid \mathsf{Mcr}] \\ \mathsf{Italy} &= \mathsf{Italy}\,[\mathsf{FIAT} \mid \mathsf{IBM} \mid \mathsf{Turin}] \end{aligned}$$

How the topology of the net looks like at this level of abstraction can be seen in Figure 3.2. Let's take a look at US. It contains four administrative domains, that intuitively correspond to the Massachusetts Institute of Technology, Stanford University, IBM and Microsoft. Each of these administrative domains has a similar structure as regards the organisation of the e-mail. Each domain contains two *users* (for simplicity we keep the number of users to two) and there is a *router* which takes the mail from one users and delivers it to another user. Any administrative domain has the following structure:

$$AD(\text{name, user}_1\text{,user}_2\text{,router}) = \text{name [user}_1 \mid \text{user}_2 \mid \text{!router]}$$

Let's take now the administrative domain of Imperial College in the UK.

$$\mathsf{IC} = AD\langle \mathsf{IC}, \mathsf{maria}, \mathsf{iain}, \mathit{Icrouter} \rangle$$

where *icrouter* and the user will be discussed later.

More specifically the administrative domain IC contains the folders of the users and a router *Icrouter*.

$$\mathsf{IC}\,[\mathsf{maria}\,[\,] \mid \mathsf{iain}\,[\,] \ldots \mid !\mathit{Icrouter}]$$

The folder of the user is an ambient whose names is the identity of the user. We can assume that this name is unique. The router is a replicated ambient, that enters e-mails and delivers them to the appropriate destination.

In following sections, we are going to describe the user and the router.

How to send an e-mail

We are going to describe the way that the user maria sends the mail. This should be more or less the same in any other domain. The mail that a user writes for sending out is composed of three parts:

i) the address of the destination for the ambient. This is an ambient called *address* which contains the address;

ii) a button that says that this mail has to be sent;

iii) the message itself that we could think as plain text , etc.

Here is how it looks the when user maria is going to send an e-mail to a friend in US.

maria[
  mail[
    open *icrouter* | *address* [⌈$sc@MIT.US$⌉] | Msg |
    send[Send]
    ]
  ]

The address is defined as follows:

$$\lceil sc@MIT.US \rceil \stackrel{def}{=} nat\,[\langle US \rangle \mid dom\,[\langle MIT \rangle \mid usr\,[\langle sc \rangle]]].$$

The address contains three kinds of information:

i) the *country* of the destination. The name of the country is contained in an ambient called *nat.*

ii) the *domain* where the possible recipient is located. The name of the domain is contained in an ambient called *dom.*

iii) the *user* for whom the mail is meant. The name of the user is contained in an ambient called *usr.*

Moreover sending an e-mail informs the router that there is an email ready to be posted:

$$\text{Send} \stackrel{def}{=} \text{out}\,mail.\text{out}\,\text{maria}.\text{in}\,icrouter.\langle \text{maria} \rangle$$

The router

We are now going to describe the router. The router is a replicated ambient that contains a program label to decode the address of the mail, and to yield to the mail the correct path to follow for the e-mail to be delivered. We assume for simplicity that it is never the case that two mails try to use the router at the same time.

$$Icrouter \stackrel{def}{=} !icrouter[\text{Button}]$$

First of all the router is activated, only when there is mail that needs to be sent.

The following process opens the ambient *send* that represent the button for sending the mail. With this process, the router gets into the user's mail. The process is the following:

$$\text{Button} \stackrel{def}{=} \text{open}\,send.(s).\text{in}\,s.\text{in}\,mail.\text{Path}$$

Next, we need to describe how the router delivers the mail. The basic idea is that the router reads the address written.

The router checks first to which countries the mail should go. If it is inside the UK, then it checks whether the domain is IC itself, in which case it delivers the mail to the user without exiting the IC domain. Otherwise, it goes outside the IC domain and enters the right domain.

If the mail is meant to go outside UK, then the router goes outside IC and UK and follows the given path. Thus, the matching operator is essential for modeling this situation.

$$\text{Path} \stackrel{def}{=} \text{open } \textit{address}.\text{open } \textit{nat}.(y).(\mathrm{D}_{uk} \mid \mathrm{D}_{us} \mid \mathrm{D}_{it})$$

If the mail was meant to be in the UK:

$$\begin{aligned}\mathrm{D}_{uk} \stackrel{def}{=} [c = UK]\text{open } \textit{dom}.(d).\quad & ([d = \mathsf{IC}]\text{out } s.\text{open } \textit{usr}.(u).\text{in } u \mid \\ & [d = \mathsf{Oxf}]\text{out } s.\text{out } \mathsf{IC}.\text{in } \mathsf{Oxf}.\text{open } \textit{usr}.(u).\text{in } u \mid \\ & [d = \mathsf{Mcr}]\text{out } s.\text{out } \mathsf{IC}.\text{in } \mathsf{Mcr}.\text{open } \textit{usr}.(u).\text{in } u))\end{aligned}$$

Otherwise the mail is going outside the UK, therefore the router is programmed to find the right path. For US the program is the following:

$$\begin{aligned}\mathrm{D}_{us} &\stackrel{def}{=} [c = \mathsf{US}]\text{out } d.\text{out } \mathsf{IC}.\text{out } \mathsf{UK}.\text{in } \mathsf{US}.\text{open } \textit{dom}.(d).R \\ R &\stackrel{def}{=} [d = \mathsf{MIT}]\text{in } \mathsf{MIT}.\text{open } \textit{usr}.(u).\text{in } u \\ &\quad\ [d = \mathsf{IBM}]\text{in } \mathsf{IBM}.\text{open } \textit{usr}.(u).\text{in } u \\ &\quad\ [d = \mathsf{Stanf}]\text{in } \mathsf{Stanf}.\text{open } \textit{usr}.(u).\text{in } u \\ &\quad\ [d = \mathsf{Mcr}]\text{in } \mathsf{IBM}.\text{open } \textit{usr}.(u).\text{in } u\end{aligned}$$

For Italy the program of the router is the following:

$$\begin{aligned}\mathrm{D}_{it} &\stackrel{def}{=} [c = \mathsf{Italy}]\text{out } s\text{out } \mathsf{IC}.\text{out } \mathsf{UK}.\text{in } \mathsf{Italy}.\text{open } \textit{dom}.(d).S \\ S &\stackrel{def}{=} [d = \mathsf{Turin}]\text{in } \mathsf{Turin}.\text{open } \textit{usr}.(u).\text{in } u \\ &\quad\ [d = \mathsf{FIAT}]\text{in } \mathsf{FIAT}.\text{open } \textit{usr}.(u).\text{in } u \\ &\quad\ [d = \mathsf{IBM}]\text{in } \mathsf{IBM}.\text{open } \textit{usr}.(u).\text{in } u\end{aligned}$$

We shall show in Figure 3.3 the interaction between the user maria, who is willing to send an e-mail to a user called sc in MIT in the US, and the local router *Icrouter*.

Concluding, we have presented here an extended example of sending an e-mail. It must be noted that if the address written by the user is wrong, the router will deadlock. Semantically this is correct, since an e-mail to a wrong address should not be delivered. However, in real life, it is common to receive unintended mail. We leave for future work to formally prove that an e-mail, written with a correct address, will be delivered to the right user assuming that none of the programs in the network interferes with the actions of the router.

IC[iain[R] | *Icrouter* |
maria[*mail*[open *icrouter* | *address* [⌈$sc@MIT.US$⌉] | Msg |
send[Send]]]] $\longrightarrow_a \ldots \longrightarrow_a$

IC[iain[R] | *Icrouter* | *icrouter* [*send*[⟨maria⟩] | Button] |
maria[*mail*[open *icrouter* | *address* [⌈$sc@MIT.US$⌉] |
Msg]]] $\longrightarrow_a \ldots \longrightarrow_a$

IC[iain[R] | *Icrouter* | *icrouter* [in maria.in *mail*.Path] |
maria[*mail*[open *icrouter* | *address* [⌈$sc@MIT.US$⌉] |
Msg]]] $\longrightarrow_a \ldots \longrightarrow_a$

IC[iain[R] | *Icrouter* |
maria[*mail*[open *address*.open *nat.*(y).(D$_{uk}$ | D$_{us}$ | D$_{it}$ | *address* [⌈$sc@MIT.US$⌉] |
Msg]]] $\longrightarrow_a \ldots \longrightarrow_a$

IC[iain[R] | *Icrouter* |
maria[*mail*[open *nat.*(y).(D$_{uk}$ | D$_{us}$ | D$_{it}$ | *nat* [⟨US⟩ | *dom* [⟨MIT⟩ | *usr* [⟨sc⟩]]] |
Msg]]] $\longrightarrow_a \ldots \longrightarrow_a$

IC[iain[R] | *Icrouter* |
maria[*mail*[(D$_{uk}\{US/y\}$ | D$_{us}\{US/y\}$ | D$_{it}\{US/y\}$) | *dom* [⟨MIT⟩ | *usr* [⟨sc⟩]] |
Msg]]] $\longrightarrow_a \ldots \longrightarrow_a$

IC[iain[R] | *Icrouter* |
maria[ ]]
mail[(D$_{uk}\{US/y\}$ | open *dom.*$(d).R$ | D$_{it}\{US/y\}$) | *dom* [⟨MIT⟩ | *usr* [⟨sc⟩]] |
Msg] $\longrightarrow_a \ldots \longrightarrow_a$

Figure 3.3: Example of sending an e-mail

Chapter 4

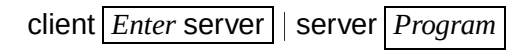
The Push and Pull Ambient Calculus

In this chapter we shall introduce the Push and Pull Ambient Calculus (PAC). In comparison with the Mobile Ambients (MA) of Cardelli and Gordon [21], two new capabilities are introduced: $\mathsf{push}\, n$ and $\mathsf{pull}\, n$ instead of $\mathsf{in}\, n$ and $\mathsf{out}\, n$; the rest of the syntax remains unchanged. We believe that this calculus can be useful for representing real-life situations, where control on the boundary of the ambients is crucial. One example of this kind of situation is the virtual learning centre, presented in Section 4.3.1. We shall see that many programming examples found in [24] can be easily adapted in PAC. Moreover, the matching operator is encodable in PAC, with identical properties to MA. Cardelli and Gordon encode the asynchronous π-calculus in MA; their encoding seems to respect some weak conditions such as preservation of the execution steps, and syntactical condition on the free names. We show that our encoding of the asynchronous π-calculus in PAC is well behaved, in the sense that it respects strong conditions such as operational correspondence and adequacy. On the other hand, the encoding is not complete, and thus full abstraction is not achieved.

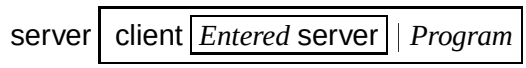
4.1 *Motivation*

The main difference between PAC and MA lies in the control of the movements of ambients. In MA, ambients are seen as autonomous agents that can enter and exit other administrative domains, if equipped with the right capability. Ambients themselves, however, are not in control of their own boundaries, thus other ambients, external or internal ones, might freely cross their boundaries. In the following example we show that an ambient called client, willing to enter into a server server,

gain access if in possession of the name of the server can easily.

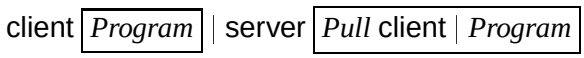


reduces to

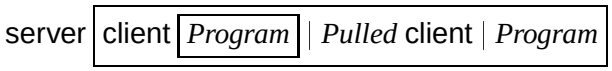


Notice that in this setting, not only client decides to gain access to server, but server cannot send client away, even in the most unsafe case in which client were a malicious agent. Similarly for exit. On the contrary, in the PAC setting, administrative domains have total control of their own boundaries, in the sense that an ambient can pull in other ambients and push away unwanted or dangerous agents.

The scenario above would be rewritten as follows:



reduces to



The semantics of PAC is similar to standard MA, but in this case the *control* of movements of the ambient lies in the server and not in the client. There does not seem to exist any obvious encoding from PAC into MA or vice versa.

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

4.1.1 *The syntax*

We assume there is an infinite set of names $\mathcal{N}$, ranged over by $n, m, \ldots$.

DEFINITION 4.1.1 The set of process terms of the push and pull ambient calculus $\mathcal{P}r_p$ is given by the following grammar:

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

M stands for the capabilities defined by the following grammar:

$$\begin{array}{rcll} M & ::= & \mathsf{pull}\, n & \textit{pull capability} \\ & \mid & \mathsf{push}\, n & \textit{push capability} \\ & \mid & \mathsf{open}\, n & \textit{open capability} \end{array}$$

We shall omit the intuitive explanation for the operators that are common to MA. See Section 2.2.1 for further explanations of these operators. As regards the capabilities M, the meaning is intuitively the following: the capability $\mathsf{pull}\, n$ causes an ambient with name n to be pulled inside the current one; the capability $\mathsf{push}\, n$ pushes an ambient with name n out of the current ambient; $\mathsf{open}\, n$ behaves as in standard MA. The capability open is necessary for PAC, as much as in MA, to allow the exchange of messages between different ambients. For example, consider the process $z\,[(r)\mathsf{pull}\, n.\mathsf{open}\, n] \mid n\,[\langle r \rangle]$. If the ambient z could not open the ambient named n, then there would not be exchange of messages. The set of free names of P, written $fn(P)$, and bound names, $bn(P)$, are defined in the standard way, taking into account that the binding operators are restriction and input. The detailed definition is similar to Definition 2.2.2. A similar observation applies to substitution, whose definition would be nearly identical to Definition 2.2.3 modulo the syntactical differences. For this reason we feel free to omit these two definitions.

4.2 *Operational semantics*

Similarly to MA, PAC is also equipped with an operational semantics defined in terms of reduction semantics.

DEFINITION 4.2.1 The *structural congruence relation* is the smallest congruence over $\mathcal{P}r_p$ that satisfies the equations in Definition 2.2.5.

DEFINITION 4.2.2 The *reduction relation* $\longrightarrow_p \subseteq \mathcal{P}r_p \times \mathcal{P}r_p$ is the smallest relation satisfying the following set of rules:

$$\begin{array}{rcll} m\,[\mathsf{pull}\, n.P \mid Q] \mid n\,[R] & \longrightarrow_p & m\,[P \mid Q \mid n\,[R]] & \text{RED PULL} \\ n\,[\mathsf{push}\, m.P \mid m\,[Q] \mid R] & \longrightarrow_p & n\,[P \mid R] \mid m\,[Q] & \text{RED PUSH} \\ \mathsf{open}\, n.Q \mid n\,[R] & \longrightarrow_p & Q \mid R & \text{RED OPEN} \\ \langle q \rangle \mid (n).P & \longrightarrow_p & P\{q/n\} & \text{RED COMM} \end{array}$$

$$\frac{P \longrightarrow_p P'}{P \mid R \longrightarrow_p P' \mid R}\ \text{RED PAR} \qquad \frac{P \longrightarrow_p P'}{(\nu n)P \longrightarrow_p (\nu n)P'}\ \text{RED RESTR}$$

$$\frac{P \longrightarrow_p P'}{n\,[P] \longrightarrow_p n\,[P']}\ \text{RED AMB} \qquad \frac{P \equiv_p P' \longrightarrow_p Q' \equiv_p Q}{P \longrightarrow_p Q}\ \text{RED CONG}$$

For PAC the canonical observable is the name of an ambient at top level, identical to MA [21, 24]. Thus, the predicate $P \downarrow n$ intuitively says that in P there is a top level process which is an ambient, whose unrestricted name is n.

DEFINITION 4.2.3 (*Strong barbs*) A process P *exhibits an ambient-barb* n, written as $P \downarrow n$, if and only if $P \equiv_a (\nu p_1 \ldots p_n)(n\,[P'] \mid P'')$, for some P', P'' and $n \notin \{p_1 \ldots p_n\}$.

Hence, by means of the framework described in Chapter 1 we formally define PAC as:

$$\mathsf{PAC} \stackrel{def}{=} \langle \mathcal{P}r_p,\ \longrightarrow_p, \downarrow \rangle.$$

We write PAC^* for the pure calculus without the communication primitives. For the definition of suitable equivalences, the notion of context is needed. This can be derived by the grammar presented in Definition 4.1.1. Finally we can present the definition of the equivalences as in the spirit of Chapter 1.

DEFINITION 4.2.4

- We write $\simeq_p$ for contextual equivalence defined over PAC as in Definition 1.3.2.
- We write $\approx_p$ for contextual bisimilarity defined over PAC as in Definition 1.3.11.

4.3 *Examples*

In this section we shall present some examples for PAC, which show the expressiveness of the calculus. The example of the learning centre (§4.3.1) shows that PAC is useful for programming resource-sensitive situations. The other examples are taken and adapted from [24].

4.3.1 *The learning centre*

In this section we seek to model resource-aware situations that might appear over the Internet, that we believe are better modelled with PAC than MA. The problem that we would like to address is a virtual learning centre LC. The town of Brighton has a virtual learning centre for students wanting to revise for their exams. Because public spending is limited, the local government can afford only a small server with a limited capacity. This means that a very limited number of students can

access the server at the same time. The council aims to apply a *fair policy*, which means that each user can stay a maximum amount of time, to avoid some users starving the service.

We aim to keep our example small and readable; thus we shall program a server that allows three users only. Each user is made out of two ambients:

i) One ambient called *request*, that simulates the request of accessing the service. Inside there is a message that contains the name of the user.

ii) Another ambient which stands for the user, that will actually enter the server and use the facilities.

The server of the council is an ambient named *server* with a program inside that allows access to at most three users at the same time. Each user is forced out of the server, to allow a new request from another user to be processed.

$$\begin{aligned} \mathsf{user}_i &\stackrel{def}{=} \mathit{request}\,[\langle \mathit{name}_i \rangle] \mid \mathit{name}_i\,[U] \\ P_j &\stackrel{def}{=} (\nu z)(z\,[\,] \mid !\mathsf{open}\ z.\mathsf{pull}\ \mathit{request}.\mathsf{open}\ \mathit{request}.(x).\mathsf{pull}\ x.\mathsf{push}\ x.z\,[\,]) \\ \mathsf{council\text{-}server} &\stackrel{def}{=} \mathit{server}[P_1 \mid P_2 \mid P_3] \end{aligned}$$

The server above contains three programs P_i, each of them deals with one user only, and each of them implements a simple form of recursion. If three users are already accessing the server, then no-one else can enter it. We believe that this kind of example would not have been possible in MA. In fact, ambients there are *not* in control of their own boundaries. Let's consider the ambient *server* $[S]$, where S is a program running inside the ambient. Because of the semantics of MA, any number of ambients that know the name *server* can have access to *server* $[S]$. One way to get round this is to restrict *server*, and take another ambient called *agent*, that gets out of *server* collects the user and goes back into the server. This is the philosophy of the example 'Agent crossing the firewall' in [24]. There are two main drawbacks in this solution that PAC eliminates.

i) The ambient *agent* is not in control of its own boundaries and it does not know when the user has entered it. Thus it might go back to *server* without any user. Moreover, if the name is public, many other ambients can enter *agent* $[\,]$.

ii) Assuming that the agent takes back to the server one user, once it has been released into the server, there is *no way* in which either the server or the agent can send away an unwanted user. As regards our example, this would compromise the constraint on fairness.

This last observation applies to SA as well. In fact, while the co-capabilities enforce a control on the boundaries, there still is no way of sending away an unwanted ambient.

The learning centre system LC is defined as follows:

$$\mathsf{LC} \stackrel{def}{=} \prod_i^4 \mathsf{user}_i \mid \mathsf{council\text{-}server}$$

4.3.2 *The encoding of choice*

We shall going to encode choice, which is going to be useful for the numerals. This was done by Cardelli and Gordon, and ours is a modification of what is present in [21].

$$\begin{aligned} n \Rightarrow P + m \Rightarrow Q \quad \stackrel{def}{=} \quad & (\nu p_1\, p_2\, q\, r\, s\, \mathrm{ok}) \\ & (p_1[\\ & \mathsf{pull}\, n.(\mathsf{push}\, s \mid s\, [\mathsf{open}\, p_2.\mathrm{ok}\, [\,]])\ \mid \\ & \mathsf{pull}\, \mathrm{ok}.\mathsf{open}\, \mathrm{ok}.\mathsf{push}\, n.(\mathsf{push}\, q \mid q\, [\mathsf{open}\, r.P])\ ]\ \mid \\ & (p_2[\\ & \mathsf{pull}\, m.(\mathsf{push}\, s \mid s\, [\mathsf{open}\, p_1.\mathrm{ok}\, [\,]])\ \mid \\ & \mathsf{pull}\, \mathrm{ok}.\mathsf{open}\, \mathrm{ok}.\mathsf{push}\, m.(\mathsf{push}\, q \mid q\, [\mathsf{open}\, r.Q])\ ]\ \mid \\ & \mathsf{open}\, s.\mathsf{open}\, q \mid r\, [\,]) \end{aligned}$$

In the presence of $n\,[R]$ and assuming that $fn(R) \cap \{p_1, p_2, q, r, s, \mathrm{ok}\} = \emptyset$ we have that:

$$(n \Rightarrow P + m \Rightarrow Q) \mid n\,[R] \quad \longrightarrow\!\!\!\!\!\rightarrow_p \simeq_p \quad P \mid n\,[R]$$

4.3.3 *Numerals*

The idea here is similar to the one that has inspired Cardelli and Gordon, so each numeral i is represented as a nested ambient of depth i, as follows:

$$\begin{aligned} \underline{0} \quad & \stackrel{def}{=} \quad zero\,[\,] \\ \underline{i+1} \quad & \stackrel{def}{=} \quad succ\,[\underline{i}] \end{aligned}$$

Now we are going to encode the increment of a numeral:

$$\begin{aligned} \mathrm{incsucc}.P \quad & \stackrel{def}{=} \quad (\nu r)(\mathsf{open}\, r.P \mid succ\,[\mathsf{pull}\, succ.(\mathsf{push}\, r \mid r\,[\,]])) \\ \mathrm{inczero}.P \quad & \stackrel{def}{=} \quad \mathsf{open}\, zero.(\underline{1} \mid P) \\ \mathrm{ifzero}P\ Q \quad & \stackrel{def}{=} \quad \mathrm{zero} \Rightarrow P + \mathrm{succ} \Rightarrow Q \\ \mathrm{inc}.P \quad & \stackrel{def}{=} \quad \mathrm{ifzero}(\mathrm{inczero}.P)(\mathrm{incsucc}.P) \\ \mathrm{dec}.P \quad & \stackrel{def}{=} \quad \mathsf{open}\, succ.P \end{aligned}$$

With the previous definition the arithmetical operations are the following:

$$\begin{aligned} \mathrm{inc}.P \mid \underline{i} \quad & \longrightarrow\!\!\!\!\!\rightarrow_p \simeq_p \quad P \mid \underline{i+1} \\ \mathrm{dec}.P \mid \underline{i+1} \quad & \longrightarrow\!\!\!\!\!\rightarrow_p \simeq_p \quad P \mid \underline{i} \end{aligned}$$

4.3.4 *Encoding of matching*

In PAC, as well as in MA it is possible to encode the matching operator in a compositional fashion. We will not report here all the details of the encoding, but only the parts that differ from

Section 3.3. If the matching operator were a primitive operator of PAC, it would yield a different language $\mathcal{P}r_{p^=}$. The reduction semantics would change by adding the following rule to structural equivalence:

$$[n=n]P \quad \equiv_p^= \quad P$$

DEFINITION 4.3.1 The encoding of matching is the function $[\![\ ]\!] : \mathcal{P}r_{p^=} \rightarrow \mathcal{P}r_p$ defined as follows:

$$[\![[m=n]P]\!] \stackrel{def}{=} (\nu z\, y)(z\,[\mathsf{open}\, m.(\mathsf{push}\, y \mid y\,[\,]) \mid n\,[\,]] \mid \mathsf{open}\, y.\mathsf{open}\, z.[\![P]\!])$$

where $[\![\ ]\!]$ for the other operators is defined compositionally or homorphically.

Notice that also in PAC, the matching operator would be encodable in the basic calculus without communication primitives.

We report below only the result on full abstraction. How to prove this theorem is shown in Section 3.3 with insignificant modifications.

PROPOSITION 4.3.2 $[\![P]\!] \lesssim [\![[n=n]P]\!]$.

Proof. See proof of Proposition 3.3.3((*iii*)).

PROPOSITION 4.3.3 (*Full abstraction*)
For all $P, Q \in \mathcal{P}r_{p^=}$, $[\![P]\!] \approx_p [\![Q]\!]$ if and only if $P \approx_p^= Q$.

Proof. See proof of Theorem 3.3.16.

The encoding above is very similar to the encoding in MA, with the difference of the use of push instead of out. First of all, neither the capability in for MA, nor the capability pull for PAC are needed. Yet, push and out capabilities are quite different. The out capability refers to the parents ambient that is going to be left. For example in the following process, $z\,[n\,[\mathsf{out}\, z]]$ the ambient n can leave the parent ambient, only if it knows its name z. This is quite different from PAC. In fact, the previous example could be adapted in the following way $z\,[\mathsf{push}\, n \mid n\,[]]$. Thus the ambient n does not have to know the name of the parent ambient in order to leave. The difference between push and out does not seem to make a difference in the encoding of the matching, since identical properties hold for both encodings. In Chapter 3 we have seen that due to the encoding of the matching, MA without communication primitives cannot be encoded into the mixed choice π-calculus without matching. Thus, since the matching operator is encodable in PAC, the latter result could be easily adapted.

THEOREM 4.3.4 There does not exist a substitution-preserving and weak barb-respecting encoding from PAC* into π_m.

Proof. The proof is similar to the proof of Theorem 3.5.1 the using the process

$$P \stackrel{def}{=} (\nu\, r)(r\,[\mathsf{push}\, n \mid m\,[\,]])$$

Moreover, in Remark 3.5.2 we observed that the minimal calculus necessary for this result was MA with the out capability only. As the process in the proof of Theorem 4.3.4 shows, a specular result holds for PAC. In other words, the smallest calculus that cannot be encoded into the mixed choice π-calculus without matching contains the push capability only. The pull capability works in a similar way to the in. The main difference lies in the control of boundaries. With the in capability, ambients can freely enters siblings. However, a public ambient being entered can never prevent its own boundaries to be crossed. With the pull capability, this is not the case. Ambients control the number and the identity of the processes crossing their own boundaries. We shall see in the next section, that this difference between capabilities influences the encoding of the asynchronous π-calculus.

4.4 *Encoding of the asynchronous π-calculus*

In this section we introduce the encoding of the asynchronous π-calculus with matching operator into PAC and present its properties. The asynchronous π-calculus was originally encoded into MA by Cardelli and Gordon in their seminal paper on Mobile Ambients [21, 23]. In a following paper on types [22] a new encoding was given together with a typed version of the previous one. For those encodings, very few properties have been proved, and none of these encodings takes into account the matching operator. The idea of the encoding of the asynchronous π-calculus into MA by Cardelli and Gordon [21] is that for each communication channel m, an ambient named m is created. Processes wishing to either input or output on channel m enter ambient m, where they communicate. Then the input process leaves m in order to continue its execution. In order to effect this, both input and output processes need to be encased in an ambient name io which is known to the channel ambient, so that they can be opened up and enabled to communicate. Thus the encoding requires a new public name. For this reason, the property of barb reflection would not hold in this setting.

Levi and Sangiorgi also provide an encoding of the asynchronous π-calculus into SA [48]. which is quite well behaved. Their results depend on the presence of co-actions. Input on channel

m is translated as an ambient m which contains an input inside. Similarly for the output. Then, for the first step of reduction, either the ambient that contains the input enters in to the one that contains the output or vice-versa. Because of co-actions, only one input (or output) is allowed to enter. Thus, concurrent redexes do not interfere with the protocols, as it would happen in standard MA. In the second step of reduction of the encoding, the ambient inside (which contains the input) gets dissolved, and communication takes place; the process continues the execution in the expected manner. For this translation, the properties of barb reflection and preservation are proved, together with the operational correspondence. However, neither the adequacy nor the completeness of the encoding with respect to barbed congruence are proved.

Our encoding translates an output on channel m into an ambient named m which contains a value. These output ambients exist in the 'ether' represented by the top level in the tree of ambients. Communication occurs when an output ambient is pulled in by a restricted ambient representing an input on m. It then dissolves its own enclosing ambient before continuing its execution. It will be the case that barbs on outputs (which are what are normally considered in the asynchronous π-calculus) correspond exactly to barbs for ambients. The encoding seems simpler than the one in [21]. There is a substantial difference in the encoding of the asynchronous π-calculus into MA and into PAC, which lies in the ability to control boundaries. The primitives of PAC enable ambients with a public name to be entered by one agent only. This is in general not possible in the MA world, and it seems that this fact causes more complications for the encoding.

DEFINITION 4.4.1 The encoding of the synchronous π-calculus into PAC is a function $\langle\!\langle\ \rangle\!\rangle : \mathcal{P}r_{\pi_a^=} \longrightarrow \mathcal{P}r_p$ defined as follows:

$$
\begin{array}{rcl}
\langle\!\langle \mathbf{0} \rangle\!\rangle & \stackrel{def}{=} & \mathbf{0} \\
\langle\!\langle m(x).P \rangle\!\rangle & \stackrel{def}{=} & (\nu lr)(\mathsf{open}\, l \mid r\,[\mathsf{pull}\, m.\mathsf{open}\, m.(x).(l\,[\mathsf{open}\, r.\langle\!\langle P \rangle\!\rangle] \mid \mathsf{push}\, l)]) \\
\langle\!\langle \overline{m}\langle q \rangle \rangle\!\rangle & \stackrel{def}{=} & m\,[\langle q \rangle] \\
\langle\!\langle (\nu n)P \rangle\!\rangle & \stackrel{def}{=} & (\nu n)\langle\!\langle P \rangle\!\rangle \\
\langle\!\langle P \mid Q \rangle\!\rangle & \stackrel{def}{=} & \langle\!\langle P \rangle\!\rangle \mid \langle\!\langle Q \rangle\!\rangle \\
\langle\!\langle [n=m]P \rangle\!\rangle & \stackrel{def}{=} & [\![[n=m]\langle\!\langle P \rangle\!\rangle]\!] \\
\langle\!\langle !P \rangle\!\rangle & \stackrel{def}{=} & !\langle\!\langle P \rangle\!\rangle
\end{array}
$$

where $l, r \notin fn(\langle\!\langle P \rangle\!\rangle)$.

NOTATION 4.4.2 The notation $[\![[n=m]\langle\!\langle P \rangle\!\rangle]\!]$ needs some explanation. We have used implicitly a composition of encodings. Strictly speaking the encoding $\langle\!\langle\ \rangle\!\rangle$ goes from $\mathcal{P}r_{\pi_a^=} \rightarrow \mathcal{P}r_{p^=}$ and preserves the matching operator, i.e. $\langle\!\langle [m=n] \rangle\!\rangle P \stackrel{def}{=} [m=n]\langle\!\langle P \rangle\!\rangle$. Then the encoding of the matching is applied as in Definition 4.3.1 from $\mathcal{P}r_{p^=}$ to $\mathcal{P}r_p$. Thus, $[\![[n=m]\langle\!\langle P \rangle\!\rangle]\!]$ stands for

the composition of two functions: the encoding of the asynchronous π-calculus $\langle\!\langle\ \rangle\!\rangle$ applies first and then encoding of the matching $[\![]\!]$. We prefer this short-hand to avoid complication in the definition of the encoding of the asynchronous π-calculus with details that are not significant.

Note that each π reduction without the matching operator is encoded by at most six steps. The first of these steps pulls the output inside the input, and commits to the π reduction. The remaining steps then proceed in a deterministic fashion, and cannot be interfered with, since they take place inside restriction.

PROPOSITION 4.4.3 $\langle\!\langle \mathcal{P}r_{\pi_a^=} \rangle\!\rangle \subset \mathcal{P}r_p$.

The inclusion is strict; it suffices to observe that the process $a\ [\ ]$ is not in the image of $\langle\!\langle\ \rangle\!\rangle$.

PROPOSITION 4.4.4 Let $P \in \mathcal{P}r_{\pi_a^=}$. Then $fn(P) = fn(\langle\!\langle P \rangle\!\rangle)$.

Proof. The proof goes by a routine induction. We present here only the interesting details.

- $P = \mathbf{0}$. Trivial.
- $P = \overline{n}\langle y \rangle$. Then $fn(\langle\!\langle P \rangle\!\rangle) = \{n, y\}$, and $fn(P) = (\{n, y\} \cup fn(P'))$.
- $P = m(x).P'$. Therefore it is the case that:

$$\begin{aligned} fn(\langle\!\langle P \rangle\!\rangle) &= \\ &\quad fn((\nu lr)(\mathsf{open}\, l \mid r\ [\mathsf{pull}\, m.\mathsf{open}\, m.(x).(l\ [\mathsf{open}\, r.\langle\!\langle P' \rangle\!\rangle]\mid \mathsf{push}\, l)])) - \{l\, r\} \\ fn(\langle\!\langle m(x).P' \rangle\!\rangle) &= (\{m\} \cup \{r\} \cup \{l\} \cup fn(\langle\!\langle P' \rangle\!\rangle)) - \{r, l\} \\ fn(\langle\!\langle P \rangle\!\rangle) &= \{m\} \cup (fn(\langle\!\langle P' \rangle\!\rangle) - \{x\}) \text{ since } \ r, l \notin fn(\langle\!\langle P' \rangle\!\rangle) \end{aligned}$$

  By induction $fn(P') = fn(\langle\!\langle P' \rangle\!\rangle)$, hence $fn(P') - \{x\} = fn(\langle\!\langle P' \rangle\!\rangle) - \{x\}$ and $fn(P) = fn(\langle\!\langle P \rangle\!\rangle)$.
- $P = [n{=}q]P'$. Since we have that $fn([\![P]\!]) = P$, then $fn(\langle\!\langle [n{=}q]P' \rangle\!\rangle) = fn([\![[n{=}q]\langle\!\langle P' \rangle\!\rangle]\!]) = fn([n{=}q]\langle\!\langle P' \rangle\!\rangle) = \{m, q\} \cup \langle\!\langle P' \rangle\!\rangle$ which by induction gives the result.

We recall that $\equiv_\pi$ does not include the rule for matching.

PROPOSITION 4.4.5 Let $P, Q \in \mathcal{P}r_{\pi_a^=}$. If $P \equiv_\pi Q$ then $\langle\!\langle P \rangle\!\rangle \equiv_p \langle\!\langle Q \rangle\!\rangle$.

Proof. By induction. For clarity, we present here only the interesting cases. It is clear that $\equiv_p$ is an equivalence relation; thus the rules for reflexivity, symmetry and transitivity are covered. As regard for the rules of congruence, the only interesting case is the preservation of the input, which

is reported below. The other cases follow trivially from the fact that the π-calculus and MA share composition, replication and restriction. As regards the rule presented in Definition 2.1.15 we report one case only, which concerns scope extrusion.

- $P \equiv_\pi Q$ then $n(y).P \equiv_\pi n(y).Q$.

$$\begin{aligned} \langle\!\langle n(y).P \rangle\!\rangle &= (\nu lr)(\mathsf{open}\, l \mid r\,[(y).l\,[\mathsf{open}\, r.\langle\!\langle P \rangle\!\rangle]] \mid \mathsf{pull}\, m.\mathsf{open}\, m.\mathsf{push}\, l]) \\ \langle\!\langle n(y).Q \rangle\!\rangle &= (\nu lr)(\mathsf{open}\, l \mid r\,[(y).l\,[\mathsf{open}\, r.\langle\!\langle Q \rangle\!\rangle]] \mid \mathsf{pull}\, m.\mathsf{open}\, m.\mathsf{push}\, l]). \end{aligned}$$

  By induction $\langle\!\langle P \rangle\!\rangle \equiv_p \langle\!\langle Q \rangle\!\rangle$ and since $\equiv_p$ is a congruence relation, it is preserved by all the contexts, therefore $\langle\!\langle n(y).P \rangle\!\rangle \equiv_p \langle\!\langle n(y).Q \rangle\!\rangle$

- $(\nu x)(P \mid Q) \equiv_\pi (\nu x)P \mid Q$ if $x \notin \mathit{fn}(Q)$. Then we have:

$$\langle\!\langle (\nu x)(P \mid Q) \rangle\!\rangle = (\nu x)\langle\!\langle P \mid Q \rangle\!\rangle = (\nu x)\langle\!\langle P \rangle\!\rangle \mid \langle\!\langle Q \rangle\!\rangle \ .$$

  By Proposition 4.4.4 we have $x \notin \mathit{fn}(\langle\!\langle Q \rangle\!\rangle)$. Therefore by using the scope extrusion of $\equiv_p$ we have $\langle\!\langle (\nu x)(P \mid Q) \rangle\!\rangle \equiv_p \langle\!\langle (\nu x)P \mid Q \rangle\!\rangle$.

PROPOSITION 4.4.6 Let $P, Q \in \mathcal{P}r_{\pi^=_{\tilde{a}}}$. If $P \equiv_{\pi^=} Q$ then $\langle\!\langle P \rangle\!\rangle \approx_p \langle\!\langle Q \rangle\!\rangle$.

Proof. The proof follows from Proposition 4.4.5, observing that $\equiv_\pi \ \subset \ \approx_p$. For the case of the matching it follows from Proposition 4.3.2 and from the fact the symmetric closure of $\lesssim$ is contained $\approx_p$

The next proposition says that if there is a barb to be observed in the π-calculus, then the same name will eventually be observed in the encoded process.

PROPOSITION 4.4.7 Let $P \in \mathcal{P}r_{\pi^=_{\tilde{a}}}$. For all a, if $P \downarrow a$ then $\langle\!\langle P \rangle\!\rangle \Downarrow a$.

Proof. Assume $P \downarrow a$. Then $P \equiv_{\pi^=} (\nu z_1 \ldots z_n)(\overline{a}\langle y \rangle \mid P')$ with $a \notin \{z_1, \ldots, z_n\}$ by Definition 2.1.11. Thus by Lemma 4.4.6:

$$\begin{aligned} \langle\!\langle P \rangle\!\rangle &\approx_p \langle\!\langle (\nu z_1 \ldots z_n)(\overline{a}\langle y \rangle \mid P') \rangle\!\rangle \\ &= (\nu z_1 \ldots z_n)\langle\!\langle \overline{a}y \rangle\!\rangle \mid \langle\!\langle P' \rangle\!\rangle \\ &\stackrel{def}{=} (\nu z_1 \ldots z_n)a\,[\langle y \rangle] \mid \langle\!\langle P' \rangle\!\rangle. \end{aligned}$$

Hence $(\nu z_1 \ldots z_n)a\,[\langle y \rangle] \mid \langle\!\langle P' \rangle\!\rangle \downarrow a$ and $\langle\!\langle P \rangle\!\rangle \Downarrow a$.

The reverse is also true; if the a barb can be observed in the encoding then it could have been observed in the source calculus as well.

PROPOSITION 4.4.8 Let $P \in \mathcal{P}r_{\pi_{\bar{a}}^{=}}$ be a process. If $\langle\langle P \rangle\rangle \downarrow n$ then $P \downarrow n$.

Proof. By induction on P.

- $P = \mathbf{0}$. Trivial.
- $P = n(y).P'$. Trivial, since for all n, we have $\langle\langle P \rangle\rangle \not\downarrow n$.
- $P = \overline{n}\langle x \rangle$. Since $\langle\langle P \rangle\rangle = n\,[\langle x \rangle]$, obviously we have $n\,[\langle x \rangle] \downarrow n$ and $\overline{n}\langle x \rangle \downarrow n$.
- $P = P' \mid P''$. Then $\langle\langle P \rangle\rangle = \langle\langle P' \mid P'' \rangle\rangle = \langle\langle P' \rangle\rangle \mid \langle\langle P'' \rangle\rangle \downarrow n$ and by induction $P' \downarrow n$. Therefore we conclude $(P' \mid P'') \downarrow n$.
- $P = (\nu x)P'$. $\langle\langle P \rangle\rangle = \langle\langle (\nu x)P' \rangle\rangle = (\nu x)\langle\langle P' \rangle\rangle \downarrow n$ and $x \neq n$. By induction $P' \downarrow n$ and $n \neq x$, therefore $(\nu x)P' \downarrow n$.
- $P = [n{=}m]P'$. Because for all n we have $\langle\langle P \rangle\rangle \not\downarrow n$, then the proposition is trivially true.
- $P = !P'$. $\langle\langle P \rangle\rangle = !\langle\langle P' \rangle\rangle$. By induction we have $P' \downarrow n$. Therefore we conclude $!P' \downarrow n$.

The next proposition claims that substitution is preserved by the translation.

PROPOSITION 4.4.9 Let $P \in \mathcal{P}r_{\pi_{\bar{a}}^{=}}$ be a process and σ a substitution. Then $\langle\langle \sigma(P) \rangle\rangle = \sigma(\langle\langle P \rangle\rangle)$.

Proof. This is consequence of Proposition 4.4.4. Since the set of free names has not changed by the translation any substitution σ on free names behaves in the same way.

THEOREM 4.4.10 Let $P \in \mathcal{P}r_{\pi_{\bar{a}}^{=}}$ be a process. Whenever $P \longrightarrow_{\pi} P'$ then for some R, S, $\langle\langle P \rangle\rangle \approx_p S \longrightarrow\!\!\!\!\rightarrow_p R \approx_p \langle\langle P' \rangle\rangle$.

Proof. The proof goes by induction on $\longrightarrow_{\pi}$. We would like to point out that up until rule RED CONG we will silently use the fact that the identity function is an expansion $\mathsf{id} \subseteq \approx_p$. At the analysis of the rule RED CONG the role of the expansion in relation to the matching operator will become clear.

- $\overline{n}\langle y\rangle \mid n(x).P \longrightarrow_\pi P\{y/x\}$.

$$
\begin{array}{rl}
& \langle\!\langle \overline{n}y \mid n(x).P\rangle\!\rangle \\
\stackrel{def}{=} & \langle\!\langle \overline{n}y\rangle\!\rangle \mid \langle\!\langle n(x).P\rangle\!\rangle \\
\stackrel{def}{=} & n\,[\langle y\rangle] \mid (\nu lr)(\mathsf{open}\, l \mid r[(x).l\,[\mathsf{open}\, r.\langle\!\langle P\rangle\!\rangle] \mid \mathsf{pull}\, n.\mathsf{open}\, n.\mathsf{push}\, l]) \\
\longrightarrow_p & (\nu lr)(\mathsf{open}\, l \mid r[(x).l\,[\mathsf{open}\, r.\langle\!\langle P\rangle\!\rangle] \mid n\,[\langle y\rangle] \mid \mathsf{open}\, n.\mathsf{push}\, l]) \\
\longrightarrow_p & (\nu lr)(\mathsf{open}\, l \mid r[(x).l\,[\mathsf{open}\, r.\langle\!\langle P\rangle\!\rangle] \mid \langle y\rangle \mid \mathsf{push}\, l]) \\
\longrightarrow_p & (\nu lr)(\mathsf{open}\, l \mid r\,[l\,[\mathsf{open}\, r.\langle\!\langle P\rangle\!\rangle\{y/x\}] \mid \mathsf{push}\, l]) \\
\longrightarrow_p & (\nu lr)(\mathsf{open}\, l \mid l\,[\mathsf{open}\, r.\langle\!\langle P\rangle\!\rangle\{y/x\}] \mid r\,[]) \\
\longrightarrow_p & (\nu lr)(\mathsf{open}\, r.\langle\!\langle P\rangle\!\rangle\{y/x\} \mid r\,[]) \\
\longrightarrow_p & (\nu lr)(\langle\!\langle P\rangle\!\rangle\{y/x\}) \\
\equiv_p & \langle\!\langle P\rangle\!\rangle\{y/x\} \\
= & \langle\!\langle P\{y/x\}\rangle\!\rangle \qquad \text{by Proposition 4.4.9}
\end{array}
$$

- If $P \mid Q \longrightarrow_\pi P' \mid Q$ then $\langle\!\langle P \mid Q\rangle\!\rangle \longrightarrow\!\!\!\!\rightarrow_p \langle\!\langle P' \mid Q\rangle\!\rangle$. Notice that $\langle\!\langle P \mid Q\rangle\!\rangle = \langle\!\langle P\rangle\!\rangle \mid \langle\!\langle Q\rangle\!\rangle$. By induction for some S, R we have $\langle\!\langle P\rangle\!\rangle \approx_p S \longrightarrow\!\!\!\!\rightarrow_p R \approx_p \langle\!\langle P'\rangle\!\rangle$. Therefore by RED PAR we have $\langle\!\langle Q\rangle\!\rangle \mid S \longrightarrow\!\!\!\!\rightarrow_p R \mid \langle\!\langle Q\rangle\!\rangle$. Since $\langle\!\langle Q\rangle\!\rangle \mid \langle\!\langle P\rangle\!\rangle \approx_p \langle\!\langle Q\rangle\!\rangle \mid S$ and $\langle\!\langle Q\rangle\!\rangle \mid \langle\!\langle P'\rangle\!\rangle \approx_p \langle\!\langle Q\rangle\!\rangle \mid R$ the proposition is proved.
- If $(\nu x)P \longrightarrow_\pi (\nu x)P'$ then for some R, S we have $\langle\!\langle (\nu x)P\rangle\!\rangle \approx_p S \longrightarrow_\pi R \approx_p \langle\!\langle (\nu x)P'\rangle\!\rangle$. Then, similar to the previous case.
- Assume that $P \equiv_{\pi^=} P' \longrightarrow_\pi Q' \equiv_{\pi^=} Q$ then $P \longrightarrow_\pi Q$. We will prove that for some R, S, $\langle\!\langle P\rangle\!\rangle \approx_p S \longrightarrow\!\!\!\!\rightarrow_p \langle\!\langle Q\rangle\!\rangle \approx_p R$. It follows from Proposition 4.4.6 and induction.

The previous proposition does not allow us to derive a stronger statement on the transitive closure of $\longrightarrow_\pi$. The main difficulty is the presence of the equivalence $\approx_p$. The equivalence $\approx_p$ is present in the formulation of the proposition because of the symmetry of the congruence rule $[n = n]P \equiv_p P$. For the implementation of matching, it makes sense to use this rule from left to right. From right to left, the rule has no computational meaning. In order to express this fact we are going to use a reduction rule that gives a computational meaning to the matching operator. This rule was used in [33].

$$\frac{P \longrightarrow_\pi P'}{[n{=}n]P \longrightarrow_\pi P'}\ \text{RED MATCH}$$

Now, we consider the asynchronous π-calculus with this extra rule for reduction and structural congruence $\equiv_\pi$. We write $\longrightarrow_{\pi}^{=}$ for this slightly different reduction relation. Thus we define $\pi_a^{=\prime} \stackrel{def}{=} \langle \mathcal{P}r_{\pi_a^=}, \longrightarrow_{\pi}^{=}, \downarrow\rangle$. We define $\approx_{\pi_a^{=\prime}}$ to be the contextual barbed bisimilarity and $\simeq_{\pi_a^{=\prime}}$ to be the contextual equivalence over $\pi_a^{=\prime}$.

First we are going to show that the new reduction relation adds no computational meaning.

PROPOSITION 4.4.11 Let $P \in \mathcal{P}r_{\pi_a^=}$. Then:

i) If $P \longrightarrow_{\pi}^{=} P'$ then $P \longrightarrow_{\pi} P'$.

ii) If $P \longrightarrow\!\!\!\!\rightarrow_{\pi}^{=} P'$ then $P \longrightarrow\!\!\!\!\rightarrow_{\pi} P'$.

iii) If $P \longrightarrow_{\pi} P'$ then, for some R, R' such that $P \equiv_{\pi^=} R$ and $P' \equiv_{\pi^=} R'$, we have $R \longrightarrow_{\pi}^{=} R'$.

iv) If $P \longrightarrow\!\!\!\!\rightarrow_{\pi} P'$ then, for some R, R' such that $P \longrightarrow\!\!\!\!\rightarrow_{\pi}^{=} R$ and $P' \equiv_{\pi^=} R'$, we have $R \longrightarrow\!\!\!\!\rightarrow_{\pi}^{=} R'$.

Proof. Straightforward from the definitions of reduction.

Moreover the contextual barbed equivalence remains the same.

PROPOSITION 4.4.12 *i*) Let $P, Q \in \mathcal{P}r_{\pi_a^=}$. $P \approx_{\pi_a^{=\prime}} Q$ if and only if $P \approx_{\pi_a^=} Q$.

ii) Let $P, Q \in \mathcal{P}r_{\pi_a^=}$. $P \simeq_{\pi_a^{=\prime}} Q$ if and only if $P \simeq_{\pi_a^=} Q$.

Proof. For part (i) the proof is similar to that in Proposition 3.3.11. For part (ii) is a consequence of Proposition 4.4.11

%Finally we can neatly prove our correspondence proposition, which says that if a term reduces in the π-calculus then the encoding would mirror the computation by adding some more steps.

PROPOSITION 4.4.13 Let $P \in \mathcal{P}r_{\pi_a^=}$. Then:

i) whenever $P \longrightarrow_{\pi}^{=} P'$ then $\langle\!\langle P \rangle\!\rangle \longrightarrow\!\!\!\!\rightarrow_{p} \langle\!\langle P' \rangle\!\rangle$;

ii) whenever $P \longrightarrow\!\!\!\!\rightarrow_{\pi}^{=} P'$ then $\langle\!\langle P \rangle\!\rangle \longrightarrow\!\!\!\!\rightarrow_{p} \langle\!\langle P' \rangle\!\rangle$.

Proof. The first part needs induction on $\longrightarrow_{\pi}^{=}$. Most of the work has been done in Theorem 4.4.10 for all the rules except RED CONG. The latter is a consequence of Lemma 4.4.5 and the induction hypothesis. The second part needs an induction on the number of steps of $\longrightarrow\!\!\!\!\rightarrow_{\pi}^{=}$.

The following proposition show that computations on the encoding do not go astray, but reflect some sort of reduction in the source calculus.

PROPOSITION 4.4.14 Let $P \in \mathcal{P}r_{\pi_a^=}$. Then:

i) whenever $\langle\!\langle P \rangle\!\rangle \longrightarrow_{p} P'$, then for some Q, $P \longrightarrow\!\!\!\!\rightarrow_{\pi}^{=} Q$ and $P' \longrightarrow\!\!\!\!\rightarrow_{p} \langle\!\langle Q \rangle\!\rangle$;

ii) whenever $\langle\!\langle P \rangle\!\rangle \longrightarrow\!\!\!\!\rightarrow_{p} P'$, then for some Q, $P \longrightarrow\!\!\!\!\rightarrow_{\pi}^{=} Q$ and $P' \longrightarrow\!\!\!\!\rightarrow_{p} \langle\!\langle Q \rangle\!\rangle$.

Proof. *i*) By structural induction on $\longrightarrow_p$. We present here only the base cases; the other cases follow by induction.

– If $\langle\!\langle P \mid Q \rangle\!\rangle \longrightarrow_p R$, and the reduction happens as the result of a communication between the two processes. Thus $\langle\!\langle P \mid Q \rangle\!\rangle = \langle\!\langle n(x).P' \mid \overline{n}\langle y \rangle \rangle\!\rangle$ then:

$$\langle\!\langle P \rangle\!\rangle \longrightarrow_p (\nu lr)(\mathsf{open}\, l \mid r[\mathsf{open}\, n.(x).(l\ [\mathsf{open}\, r.\langle\!\langle P \rangle\!\rangle] \mid \mathsf{push}\, l) \mid n\ [\langle y \rangle]]).$$

The reduction on the π-calculus-side could be: $n(x).P' \mid \overline{n}\langle y \rangle \longrightarrow_{\pi}^{=} P'\{y/x\}$ and by following the steps of reduction

$$(\nu lr)(\mathsf{open}\, l \mid r[\mathsf{open}\, n.(x).(l\ [\mathsf{open}\, r.\langle\!\langle P \rangle\!\rangle] \mid \mathsf{push}\, l) \mid n\ [\langle y \rangle]]) \longrightarrow\!\!\!\!\rightarrow_p \langle\!\langle P\{y/x\} \rangle\!\rangle.$$

– if $\langle\!\langle P \rangle\!\rangle = [\![[n{=}m]\langle\!\langle P' \rangle\!\rangle]\!]$ then:

1) if $n \neq m$ then $\langle\!\langle P \rangle\!\rangle \not\longrightarrow_{\pi}$. Therefore the Proposition is trivially true;

2) if $n = m$ then $\langle\!\langle P \rangle\!\rangle = [\![[n{=}n]\langle\!\langle P' \rangle\!\rangle]\!]$:

$$(\nu z\, y)(z\ [\mathsf{open}\, m.(\mathsf{push}\, y \mid y\ [\,] \mid n\ [\,]] \mid \mathsf{open}\, y.\mathsf{open}\, z.\langle\!\langle P \rangle\!\rangle)) \longrightarrow_p (\nu z\, y)(z\ [(\mathsf{push}\, y \mid y\ [\,]] \mid \mathsf{open}\, y.\mathsf{open}\, z.\langle\!\langle P \rangle\!\rangle))$$

Then we have $P' \longrightarrow\!\!\!\!\rightarrow_{\pi}^{=} P'$. By following the reduction steps it is the case that

$$(\nu z\, y)(z\ [(\mathsf{push}\, y \mid y\ [\,]] \mid \mathsf{open}\, y.\mathsf{open}\, z.\langle\!\langle P \rangle\!\rangle)) \longrightarrow\!\!\!\!\rightarrow_p \langle\!\langle P' \rangle\!\rangle.$$

Thus $\langle\!\langle P' \rangle\!\rangle \longrightarrow\!\!\!\!\rightarrow_p \langle\!\langle P' \rangle\!\rangle$ and the proposition holds.

.

ii) By induction on the number of steps.

THEOREM 4.4.15 Let $PQ \in \mathcal{P}r_{\pi_a^{=}}$.

i) If $\langle\!\langle P \rangle\!\rangle \approx_p \langle\!\langle Q \rangle\!\rangle$ then $P \approx_{\pi_a^{='}} Q$.

ii) If $\langle\!\langle P \rangle\!\rangle \simeq_p \langle\!\langle Q \rangle\!\rangle$ then $P \simeq_{\pi_a^{='}} Q$.

Proof. *i*) We prove that

$$\mathcal{S} = \{(P, Q) : \langle\!\langle P \rangle\!\rangle \approx_p \langle\!\langle Q \rangle\!\rangle\}$$

is a weak contextual bisimulation over $\pi_a^{='}$.

Assume $P \downarrow n$; then by Proposition 4.4.7 we have $\langle\!\langle P \rangle\!\rangle \Downarrow n$. Therefore $\langle\!\langle Q \rangle\!\rangle \Downarrow n$, which implies by Proposition 4.4.8 that $Q \Downarrow n$. Assume now for all $\mathcal{C}$ that $\mathcal{C}[P] \longrightarrow_{\pi}^{=} P'$. Then by Proposition 4.4.13 $\langle\!\langle \mathcal{C}[P] \rangle\!\rangle \longrightarrow\!\!\!\!\rightarrow_p \langle\!\langle P' \rangle\!\rangle$, and for some Q' it is the case that $\langle\!\langle \mathcal{C}[Q] \rangle\!\rangle \longrightarrow\!\!\!\!\rightarrow_p Q'$

and $P' \approx_p Q'$. Notice that $\mathcal{C} \in \langle\langle\ \rangle\rangle$. Therefore by Proposition 4.4.14 ((*ii*)) there exist a Q' such that $\mathcal{C}[Q] \longrightarrow\!\!\!\!\!\rightarrow_{\overline{\pi}}^{=} Q'$. Thus we conclude $(P', Q') \in \mathcal{S}$ and $\mathcal{S} \approx_{\pi_{\overline{a}}^{=\prime}}$. Then by Proposition 4.4.12 our result is proved.

ii) For part one we prove that if $P \Downarrow n$ then $P \Downarrow n$ and the symmetric clause is similar. The rest of Theorem is a consequence of Proposition 4.4.13 and Proposition 4.4.14.

4.5 *Conclusions*

The encoding of the asynchronous π-calculus in PAC respects some strong conditions, such as operational correspondence, barb preservation and reflection, and adequacy. However, full abstraction, i.e. completeness of the encoding, does not hold. In fact, in the asynchronous π-calculus $m(n).\overline{m}\langle n\rangle \approx_\pi \mathbf{0}$, whereas $\langle\langle m(n).\bar{m}\langle n\rangle\rangle\rangle$ is not equivalent to $\langle\langle \mathbf{0}\rangle\rangle$ even under $\simeq_p$. This is mainly due to the presence of contexts in PAC that do not fall into the image of the encoding, and disrupt the protocol by interacting with the translation of an input. Clearly, such a problem could be eliminated by considering an equivalence on PAC defined on π-contexts only, or by devising some clever type system, that would automatically eliminate 'bad' contexts.

Our encoding is quite different from the one presented in [23] as also explained in Section 4.4. That encoding only preserves the execution steps of the asynchronous π-calculus, but the reflection of the reduction steps cannot be preserved, which means that some reductions go 'wrong' in the translation, i.e. they do not correspond to reduction steps in the asynchronous π-calculus. For this reason, equivalences in general cannot be preserved, in other words adequacy or completeness do not hold. For Levi and Sangiorgi's encoding [48] operational correspondence proposition holds, and also preservation and reflection of barbs. However, results related to adequacy are not proved, although in our view, such a results should hold.

Chapter 5

From names to sound theories

In this chapter we investigate equivalences in the reduction semantics framework by means of equational reasoning. This work has been inspired by Honda and Yoshida [46], who devised this method and applied it to various dialects of the π-calculus. They [46] have proposed a general formulation of process semantics which induces a canonical congruence based solely on the reduction relation and equational reasoning. In fact throughout this chapter we shall develop a construction of an equivalence relation that we prove coincides with contextual barbed bisimulation. One of the novelties of this approach is that we do not have to commit ourselves *a priori* to any kind of observational predicate, or *barb*. We shall see that observables are derivable in this setting unlike barbed bisimulation [62]. The main contribution of this chapter is to show that Honda and Yoshida's framework can be non-trivially generalized to MA. In [73, 87] we have shown that this construction gives interesting properties on the barbs of SA and PAC.

5.1 *Background*

The framework we are going to introduce here was developed for the asynchronous π-calculus by Honda and Yoshida [46]. The original idea was to derive a meaningful equivalence based solely on the reduction relation and equational reasoning, in the spirit of the theory for the λ-calculus [6].

Bisimulation equivalences in the concurrency setting are traditionally defined on labelled transition systems [54]. As already noticed in Chapter 1, some calculi enjoy more than one labelled transition system; thus more than one bisimulation is definable. If the different equivalences coincide, then it does not matter which one is considered. However, this is not the case for the

π-calculus.

Milner [62] argues that reduction semantics is a uniform paradigm for describing semantics for calculi that are indeed syntactically very different from each other. In fact, reduction semantics has been widely used [55, 58, 62, 76] for its simplicity and ability to represent uniformly simple process calculi such as CCS [58], and first and second order name-passing calculi such as the π-calculus and $\mathcal{HO}$-π-calculus [55, 62, 76]. Milner and Sangiorgi attempt to define reduction semantics by using co-inductive methods developed in the labelled transition framework. This means that *barbed bisimulation* relies on the definition of reduction relation and an observational predicate, or *barb* [62, 46, 76] (see also Definition 1.3.4). In higher-order calculi, the reduction relation seems to be simpler and more elegant. However with barbed bisimulation one needs to use the power of contexts to get a congruence (see Definition 1.3.7), and one must also commit a priori to a notion of barb, where in calculi such as MA there might be several possible choices.

Honda and Yoshida have different motivations for developing their framework. They seek to adapt "well-studied methods found in both strict and lazy lambda calculus [6] by using maximality conditions among a certain family of congruences to derive canonical equality over agents". Moreover the main point is that two processes considered equivalent should remain in the same class during computation.

The starting point is the definition of *insensitive terms*, which are terms that do not interact with the environment. We then define sound theories. A theory is *sound* if it identifies insensitive and structurally equivalent terms, is consistent, and is respected by contexts and by the reduction relation. We will see that these minimal requirements are enough to induce *contextual barbed bisimulation*, (as in Definition 1.3.11). In the asynchronous π-calculus setting, contextual barbed bisimulation and barbed congruence coincide [32]. In our setting it remains future work to prove this.

In the current framework we do not have to commit in advance to any kind of observation – any observable becomes a property with which the theory is equipped a posteriori. In the ambient setting, unlike the π-calculus, different possible choices for barbs have been made, for example observing entering an ambient, as remarked in Remark 2.2.9. Using the Honda-Yoshida techniques, the observables are indeed the names of the top level ambients, confirming the original choice of barbs made by Cardelli and Gordon.

5.2 *Basic construction for Mobile Ambients*

In this section we will show the basic construction of sound theories for MA. First of all, an *ambient theory*, or simply a theory $\mathcal{T}$, is an equivalence relation that is closed under all contexts

and contains at least structural congruence. As said in the introduction we seek to preserve this relation throughout computation. This induces the definition of *reduction closure* over the theory. Similarly to λ-theories, where divergent terms are put in a single equivalence class, we identify here meaningless terms, i.e. terms that are equivalent to **0** in a weak regime. These are called *insensitive terms*. Finally, we require that the theory is not trivial, in the sense that it does not equate any two terms. We call this condition *consistency*. Insensitive terms are invisible terms, in the sense that they never interact with any context. The definition of insensitive terms in the MA setting is far from being easy or trivial, but we will show that our definition, although complicated, works. The definition requires an elaborate analysis of free names of a process.

5.2.1 *Insensitive terms*

An insensitive process (or term) is one which can never react with its environment or the surrounding *context*. In order to define insensitive terms, we need to talk about special set of free names of a term: *active names*. Active names in a term P are the ones that allow P to immediately engage in a reduction with a context. In the setting of the asynchronous π-calculus channel names seem a natural choice. Thus, Honda and Yoshida [46] identify define the active names $an(P)$ of a process P:

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

(together with the clauses for replication, parallel, restriction).

It is rather harder to define active names for the Ambient Calculus. To justify the choice of active names, it is necessary to analyse the role of names in the reduction relation. The term $R = \mathsf{open}\, n.P$ may react with a context $n\,[Q]$ but *not* with $q\,[Q]$. Thus, the name n plays a fundamental role in the reduction. A term is insensitive if for all of its derivatives (including itself) the set of active names (i.e. the vehicle for engaging reduction) is empty, i.e. if for all P' such that $P \longrightarrow\!\!\!\!\!\rightarrow P'$ then $an(P') = \emptyset$. Following the work done on π-calculus, in MA we would like n to be active in $n\,[P]$ as well as $\mathsf{open}\, n.P$. We also have to allow that n is active in $m\,[\mathsf{in}\, n.P]$. Yet, anonymous communication primitives in MA are to be treated with care. In fact there is no evident name that can express that the process $(x).P$ can immediately engage in a reduction with the surrounding context. Similarly $(\nu n)\langle n\rangle$, where there are no free names. To the purpose of excluding the previous two processes, from the class of insensitive names, we introduce in the definition of active names a special name $\star$ which indicates willingness to communicate at the top level. Thus, we augment the set of names $\mathcal{N}' = \mathcal{N} \cup \{\star\}$. In this thesis, we consider only communication of names, as opposed to [21] where also capabilities can be sent. Had we taken the latter approach, the definition of active names would possibly be the same, and also the general

development of sound theories, providing that there would not be run-time errors from sending capabilities where names are expected.

DEFINITION 5.2.1 The set of *active names* of MA processes is defined as follows:

$$\begin{array}{rclcrcl}
an(\mathbf{0}) & = & \emptyset & & an((\nu n)P) & = & an(P) - \{n\} \\
an(\mathsf{in}\, n.P) & = & \{n\} & & an(P \mid Q) & = & an(P) \cup an(Q) \\
an(\mathsf{out}\, n)P & = & \{n\} & & an(!P) & = & an(P) \\
an(\mathsf{open}\, n.P) & = & \{n\} & & an((n).P) & = & \{\star\} \\
an(n\,[P]) & = & \{n\} \cup (an(P) - \{\star\}) & & an(\langle n \rangle) & = & \{\star\}
\end{array}$$

LEMMA 5.2.2 $an(P) \subseteq fn(P) \cup \{\star\}$.

Proof. By induction on P.

Below we report the definition of insensitive terms.

DEFINITION 5.2.3 (*cf [46]*) A process P is *insensitive* if $an(P') = \emptyset$ for all P' such that $P \longrightarrow\!\!\!\!\!\rightarrow_a P'$.

Notice that for the definition of insensitive terms, the choice of active names is crucial.

NOTATION 5.2.4 The set of insensitive terms is written Ins. We reserve the metavariable U and V for insensitive terms.

EXAMPLE 5.2.5 Examples of insensitive terms:

- $(\nu p_1 \ldots p_n, k)(k\,[P])$ where $fn(P) \subseteq \{p_1 \ldots p_n\}$;
- $(\nu n)(\mathsf{open}\, n.P)$;
- $(\nu n)(\mathsf{in}\, n.P)$;
- $(\nu n)(!\mathsf{open}\, n.n\,[\,] \mid n\,[\,])$;

By contrast $(\nu w)w\,[\mathsf{in}\, r.P]$ is *not* insensitive.

The following three lemmas tell us that no observations are possible in the case of insensitive processes, and that they are closed under structural congruence. We will need the following lemma for the next proof.

LEMMA 5.2.6 *i*) If $P \equiv_a Q$ then $fn(P) = fn(Q)$.

ii) If $P \equiv_a Q$ then $an(P) = an(Q)$.

Proof. *i*) The proof is basically as part (*ii*).

ii) The proof goes by induction on the definition of structural congruence. We present here only the interesting cases that have to deal with restriction. The other cases follow easily from the fact that $an(P)$ is compositionally defined.

- If $(\nu nm)P \equiv_a (\nu mn)P$ then $an((\nu nm)P') = an((\nu mn)P)$. $an((\nu nm)P) = (an(P) - \{n\}) - \{m\} = an(P) - \{m, n\} = (an(P) - \{m\}) - \{n\} = an((\nu mn)P)$.
- If $(\nu n)(P \mid Q) \equiv_a (\nu n)P \mid Q$ then $an((\nu n)(P \mid Q)) = an((\nu n)P \mid Q)$ if $n \notin fn(Q)$. Since $an(P) \subseteq fn(Q)$ then $n \notin an(Q)$. $an((\nu n)(P \mid Q)) = (an(P) \cup an(Q)) - \{n\} = (an(P) - \{n\}) \cup an(Q) = an((\nu n)P \mid Q))$.
- If $(\nu n)(m\,[P]) \equiv_a (m\,[(\nu n)P])$ then $an((\nu n)(m\,[P])) = an((m\,[(\nu n)P]))$ if $n \neq m$. $an((\nu n)(m\,[P])) = an(m\,[P]) - \{n\} = (an(P) - \{\star\}) \cup \{m\}) - \{n\}$. This is equal to $((\{m\} \cup (an(P) - \{\star\}) - \{n\}) = an((m\,[(\nu n)P]))$, since $m, n, \star$ are all different.

LEMMA 5.2.7 For all $P \in \mathcal{P}r_a$, $an(\sigma(P)) = \sigma(an(P))$.

Proof. By induction on P.

- If $P = \mathbf{0}$ then trivial.
- If $P = \mathsf{open}\, n.P'$ then $an(\mathsf{open}\, n.P) = \{n\}$. Now $an\sigma(\mathsf{open}\, n.P') = an(\mathsf{open}\, \sigma(n).\sigma(P)) = \{\sigma(n)\}$. Since σ is a function the lemma is proved.
- If $P = \mathsf{out}\, n.P'$ or $P = \mathsf{in}\, n.P'$ or $P = \langle n \rangle$ then the proof is similar to the previous case.
- If $P = (y).P'$, then we extent the substitution to the name $\star$ such that $\sigma(\star) = \star$. Thus, $an((y).P') = \{\star\}$ from which follows that $\sigma(an((y).P')) = \{\star\}$; moreover observing that $\sigma((y).P') = (y).\sigma(P')$ and $an((y).\sigma(P')) = \{\star\}$ the lemma is proved.
- $P = n\,[P']$ then $an(n\,[P']) = \{n\} \cup an(P') - \{\star\}$. Consider that $\sigma(n\,[P'] = \sigma(n)\,[\sigma(P')]$ then $an(\sigma(n\,[P'])) = \{\sigma(n)\} \cup an(\sigma(P')) - \{\star\}$. By induction $an(\sigma(P')) = \sigma(an(P'))$, since $\sigma(\star) = \star$ and σ is a function, the lemma is proved.
- $P = P' \mid P''$ or $P = !P'$ the result follows by induction.

LEMMA 5.2.8 *i*) If $an(P) = \emptyset$ then for all n, $P \not\downarrow n$.

ii) If $U \in \mathsf{Ins}$ then, for all n, $U \not\Downarrow n$.

Proof. *i*) By contradiction, let's assume that $P \downarrow n$ then $P \equiv_a (\nu p_1 \ldots p_n)(n\,[P'] \mid P'')$ and

$n \notin \{p_1 \ldots p_n\}$. Then by definition of active names n is active. Thus by Lemma 5.2.6((*ii*)) $n \in an(P)$. Contradiction, P is insensitive hence $an(P) = \emptyset$.

ii) It follows from (*i*) and from Definition 5.2.3 of insensitive terms.

LEMMA 5.2.9 *i*) If $U \in \mathsf{Ins}$ and $U \equiv_a P$ then $P \in \mathsf{Ins}$.

ii) If $U \in \mathsf{Ins}$ and $U \longrightarrow\!\!\!\!\!\rightarrow_a U'$ then U' is insensitive.

Proof. *i*) Since $an(U) = \emptyset$ by definition of insensitive terms and $U \equiv_a P$ then $an(P) = \emptyset$ by Lemma 5.2.6. Moreover, by definition of $\longrightarrow_a$ P has the same derivatives as U.

ii) Follows from Definition 5.2.3.

LEMMA 5.2.10 Let σ be a substitution and P a process. If $an(P) = \emptyset$ then $an(\sigma(P)) = \emptyset$.

Proof. The result follows from Lemma 5.2.7.

PROPOSITION 5.2.11 If U is insensitive and σ is a substitution, then $\sigma(U)$ is insensitive.

Proof. The proof can be found in the next section.

The proof of Proposition 5.2.11 is long, quite differently from [46]. The reason is that in order to conclude if $\sigma(P) \longrightarrow\!\!\!\!\!\rightarrow_p \sigma(P')$ then $an(\sigma(P')) = \emptyset$, we need to show that if $\sigma(P) \longrightarrow\!\!\!\!\!\rightarrow_p \sigma(P')$ then $P \longrightarrow_a P'$, which is not true in general for non-injective substitutions. Because we like to work completely in an unlabelled setting we need to capture which terms change their semantics in the presence of non-injective substitutions.

5.3 *Reduction with respect to a name*

We introduce here the notion of *quasi-redex*, which is a notion that completely captures the sensitivity of a term under non-injective substitutions. After a number of very basic notions we will prove Proposition 5.2.11.

DEFINITION 5.3.1 A process P is said to be a *redex* if it is structurally congruent to one of the

following forms:

$$\mathsf{open}\, n.R \mid n\,[Q]$$
$$r\,[\mathsf{in}\, n.R \mid Q] \mid n\,[Z]$$
$$r\,[R \mid q\,[\mathsf{out}\, r.Z \mid T]]$$

LEMMA 5.3.2 Let $P, Q \in \mathcal{P}r_a$. If $P \equiv_a Q$ then, for all substitutions σ, $\sigma(P) \equiv_a \sigma(Q)$.

Proof. By induction on $\equiv_a$.

LEMMA 5.3.3 Let $P \in \mathcal{P}r_a$. If $P \longrightarrow_a P'$ then $\sigma(P) \longrightarrow_a \sigma(P')$.

Proof. By induction and using Lemma 5.3.2

The converse is strictly false, on the grounds that the non-injective substitutions induce new redexes; for instance $\mathsf{open}\, r \mid n\,[P]$.

There are circumstances under which the reverse will be true; we are going to show those.

LEMMA 5.3.4 Assume $Q = \langle n \rangle \mid (x).P$. If $\sigma(Q) \longrightarrow_a \sigma(P\{n/x\})$ then $Q \longrightarrow_a P\{n/x\}$.

Proof. Trivial.

More in general, in analysing the reduction of a term, without the help of a labelled transition system, we need to capture precisely the behaviour of the reduction that is insensitive to substitutions.

DEFINITION 5.3.5 *i*) A process P contains an active(m, q)quasi-redex if:

$$\begin{array}{lll} R & \equiv_a & (\mathsf{open}\, q.P \mid m\,[Q]) \mid (m\,[Q] \mid r\,[\mathsf{in}\, q.P \mid Z]) \mid (m\,[r\,[\mathsf{out}\, q.P \mid T] \mid Q]) \\ S & ::= & R \mid r\,[S] \mid Q \mid S \mid S \mid Q \mid (\nu r)S \mid\, !S \end{array}$$

with $r \neq m, q$.

ii) A process P contains an active-quasi-redex if for some m, q P contains an active(m, q)quasi-redex.

LEMMA 5.3.6 If P contains an active(m, q)quasi-redex then $n, m \in an(P)$.

Proof. By induction on the definition of active subterm.

- If $P = R$ then by syntactical inspection of R the lemma is proved.
- If $P = Q \mid S$ and S contains a active(m, q)quasi-redex, then $m, q \in an(S)$ by induction. Since $an(P) = an(Q) \cup an(S)$ then we have $m, q \in an(P)$.

- If $P = S \mid Q$ then the proof is similar to the previous case.
- If $P = r\,[S]$ and S contains an active(m, q)quasi-redex, then $m, q \in an(S)$ by induction. Since $an(P) = \{r\} \cup an(S)$ then we have $m, q \in an(P)$.
- If $P = (\nu n)S$ and S contains a active(m, q)quasi-redex, then $m, q \in an(S)$ by induction. Since $an(P) = an(S) - \{n\}$ and $n \neq m, q$ then we have $m, q \in an(P)$.
- If $P = !S$ since by definition of active names we have that $an(!P) = an(S)$, then the lemma follows by induction.

Finally we prove the most important result for this construction. If a process does not contain any active quasi-redex as subterm, then a non-injective substitution does not increase the number of reductions.

PROPOSITION 5.3.7 If P does not contain any active-quasi-redex, then for all substitutions σ if $\sigma(P) \longrightarrow_a Q$ then for some P' $P \longrightarrow_a P'$ and $\sigma(P') = Q$.

Proof. By induction on $\longrightarrow_a$.

- If $\sigma(P) \longrightarrow_a Z$ by means of the rule RED OPEN
  then $\sigma(P) = \mathsf{open}\,\sigma(r).\sigma(R) \mid \sigma(q)\,[\sigma(Q)] \longrightarrow_a Z = \sigma(R) \mid \sigma(Q) = \sigma(R \mid Q)$. Since by hypothesis P does not contain any quasi-redex, we have $r = q$. In fact, if $r \neq q$ then there would be a quasi-redex(r, q). Thus we conclude $\mathsf{open}\,q.R \mid q\,[Q] \longrightarrow_a R \mid Q$.
- If $\sigma(P) \longrightarrow_a Z$ by means of the rule RED IN or RED OUT then the proof is similar to the previous case.
- If $\sigma(P) \longrightarrow_a Z$ by means of the rule RED COMM then by Lemma 5.3.4.
- If $\sigma(P) \longrightarrow_a Z$ and last rule used was RED PAR then $\sigma(P) = \sigma(R \mid Q) = \sigma(R) \mid \sigma(Q) \longrightarrow_a Z = T \mid \sigma(Q)$. By induction we have $\sigma(R) \longrightarrow_a T$ and $R \longrightarrow_a R'$ and $\sigma(R') = T$. Thus $R \mid Q \longrightarrow_a R' \mid Q$ by an application of the rule RED PAR and $\sigma(R' \mid Q) = \sigma(R') \mid \sigma(Q)$.
- If $\sigma(P) \longrightarrow_a Z$ and last rule used was RED PAR or RED AMB or RED RESTR, then it follows by induction.
- If $\sigma(P) \longrightarrow_a Z$ and last rule used was RED CONG then it follows by observing that if P does not contain any active-quasi-redex and $P \equiv_a Q$ then Q does not contain any active-quasi-redex, and the induction hypothesis.

Now, we claim that an insensitive term does not contain any active subterm.

LEMMA 5.3.8 *i*) If $an(U) = \emptyset$ then U does not contain any active-quasi-redex.

ii) If $U \in$ Ins then U does not contain any active-quasi-redex.

Proof. *i*) Assume that U contains an active quasi-redex as subterm, then there exists a term R that is quasi-redex(m, q) such that R is an active subterm. Then by Lemma 5.3.6 $n, q \in an(U)$. Contradiction, since $an(P) = \emptyset$.

ii) It follows from part (*i*).

PROPOSITION 5.3.9 If $P \in$ Ins then if $\sigma(P) \longrightarrow\!\!\!\!\!\rightarrow_a Q$ then for some P' $P \longrightarrow\!\!\!\!\!\rightarrow_a P'$ and $\sigma(P') = Q$.

Proof. By induction on $\longrightarrow\!\!\!\!\!\rightarrow_a$ and by applying Proposition 5.3.7 and Lemma 5.3.8.

Finally we can prove Proposition 5.2.11.

Proof. Assume that $U \in$ Ins then $an(U) = \emptyset$ and by Lemma 5.2.10 $an(\sigma(U)) = \emptyset$. By Lemma 5.3.8, U does not contain any active subterm as quasi-redex. If $\sigma(U) \longrightarrow\!\!\!\!\!\rightarrow_a Q$ then by Proposition 5.3.9, $U \longrightarrow\!\!\!\!\!\rightarrow_a U'$ and $\sigma(U') = Q$. Thus $an(Q) = an(\sigma(U')) = \emptyset$ by Lemma 5.2.10 for any Q, which implies that $\sigma(U)$ is insensitive.

5.4 *From ambient theories to sound theories*

In this section we define the notion of ambient theory, reduction closure and consistency.

We shall consider equational theories, *ambient theories,* which are simply sets of pairs (P, Q) of processes, from which we can deduce equations between processes by the usual laws of equational reasoning.

DEFINITION 5.4.1 An ambient theory contains at least the following axioms or rules.

$$
\begin{array}{lll}
(1) & P = Q \quad \text{if } P \equiv_a Q & (5) \quad P = Q \Longrightarrow M.P = M.Q \\
(2) & P = Q \Longrightarrow Q = P & (6) \quad P = Q \Longrightarrow (\nu w)P = (\nu w)Q \\
(3) & P = Q \;\; Q = Z \Longrightarrow P = Z & (7) \quad P = Q \Longrightarrow n\,[P] = n\,[Q] \\
(4) & P = Q \Longrightarrow P \mid R = Q \mid R & (8) \quad P = Q \Longrightarrow (y).P = (y).Q \\
(4') & P = Q \Longrightarrow R \mid P = R \mid Q & (9) \quad P = Q \Longrightarrow \sigma(P) = \sigma(Q) \\
 & & (10) \quad P = Q \Longrightarrow !P = !Q
\end{array}
$$

The theory above can be seen as identical to structural congruence, however in general a theory might have more axioms $P = Q$.

NOTATION 5.4.2 We let $\mathcal{T}$ range over theories. We will say that $\mathcal{T} \vdash P = Q$ if $P = Q$ is derivable in $\mathcal{T}$, and $\mathcal{T} \nvdash P = Q$ otherwise. We will write $P = Q$, $P \neq Q$ when it is clear from the context which particular theory we are referring to.

We shall need to impose some conditions of *soundness* on theories in order that they capture behavioural equivalence.

These conditions include the identification of insensitive terms and *reduction closure*, meaning that equality is preserved through reduction. We will see these minimal requirements are powerful enough to define a meaningful congruence over processes.

DEFINITION 5.4.3 A theory is *reduction-closed* if whenever $P = Q$ and $P \longrightarrow\!\!\!\!\!\rightarrow_a P'$ then there exists a Q' such that $Q \longrightarrow\!\!\!\!\!\rightarrow_a Q'$ and $P' = Q'$.

PROPOSITION 5.4.4 (*cf.[46]*) *i*) A theory $\mathcal{T}$ is reduction-closed if and only if whenever $\mathcal{T} \vdash P = Q$ then for all contexts $\mathcal{C}$, $\mathcal{C}[P] \longrightarrow\!\!\!\!\!\rightarrow_a P'$ implies for some Q', $\mathcal{C}[Q] \longrightarrow\!\!\!\!\!\rightarrow_a Q'$ with $\mathcal{T} \vdash P' = Q'$.

ii) A theory $\mathcal{T}$ is reduction-closed if and only if whenever $\mathcal{T} \vdash P = Q$ $P \longrightarrow_a P'$ implies for some Q', $Q \longrightarrow\!\!\!\!\!\rightarrow_a Q'$ with $\mathcal{T} \vdash P' = Q'$.

Proof. *i*) It follows from Definition 5.4.1 and Definition 5.4.3 of reduction closure.

ii) The 'only if' direction follows trivially by Definition 5.4.3 of reduction closure. For the 'if' direction we prove by induction on the number of steps of $P \longrightarrow\!\!\!\!\!\rightarrow_a P'$. We write $P \longrightarrow_a^n P'$ when $P \equiv_a P'$ if $n = 0$, else $P \longrightarrow_a^{n-1} \longrightarrow_a P'$. Thus, $P \longrightarrow\!\!\!\!\!\rightarrow_a P'$ if and only if, for some n, $P \longrightarrow_a^n P'$.

a) Assume that $\mathcal{T} \vdash P = Q$ and $P \longrightarrow\!\!\!\!\!\rightarrow_a P$ then clearly $Q \longrightarrow\!\!\!\!\!\rightarrow_a Q$.

b) Assume that $\mathcal{T} \vdash P = Q$ and $P \longrightarrow_a P$. Then by hypothesis, for some Q', $Q \longrightarrow\!\!\!\!\!\rightarrow_a Q'$ and $\mathcal{T} \vdash P' = Q'$.

c) Assume that $\mathcal{T} \vdash P = Q$ and $P \longrightarrow_a^{n-1} S \longrightarrow_a P'$ then by induction for some Q, $Q \longrightarrow\!\!\!\!\!\rightarrow_a R$ and $\mathcal{T} \vdash S = R$. The conclusion follows from the previous case.

Before going on we introduce some proof techniques for showing reduction closure. The following set proposition allows us to cut down the number of contexts necessary for defining the reduction closure.

DEFINITION 5.4.5 The set of *reduction contexts* is defined as follows:

$$\mathcal{C}_r ::= [\,] \mid (\nu n)\mathcal{C}_r \mid P \mid \mathcal{C}_r \mid \mathcal{C}_r \mid P \mid n\,[\mathcal{C}_r]$$

NOTATION 5.4.6 The definition of reduction-closed theory can be easily adapted from Definition 5.4.1 using reduction contexts only. In this case we write $\mathcal{T}^-$. In this case we write:

(1)	$P = Q$ if ..	(4′)	$P = Q \Longrightarrow R \mid P = R \mid Q$
(2)	$P = Q \Longrightarrow Q = P$	(5)	$P = Q \Longrightarrow M.P = M.Q$
(3)	$P = Q \;\; Q = Z \Longrightarrow P = Z$	(6)	$P = Q \Longrightarrow (\nu w)P = (\nu w)Q$
(4)	$P = Q \Longrightarrow P \mid R = Q \mid R$	(7)	$P = Q \Longrightarrow n\,[P] = n\,[Q]$
		(8)	$P = Q \Longrightarrow (y).P = (y).Q$

Notice that we can restore $\mathcal{T}$ simply by adding (9) and (10) from 5.4.1. The condition (1) is left specified, since is might contain any axions.

DEFINITION 5.4.7 A theory $\mathcal{T}^-$ is $\mathcal{C}_r$*-reduction-closed,* if whenever $\mathcal{T}^- \vdash P = Q$ and $P \longrightarrow\!\!\!\!\!\rightarrow_a P'$ then there exists a Q' such that $Q \longrightarrow\!\!\!\!\!\rightarrow_a Q'$ and $\mathcal{T}^- \vdash P' = Q'$.

LEMMA 5.4.8 (*cf.[46]*) Suppose that $\mathcal{T}$ is a usual ambient theory and $\mathcal{T}^-$ a $\mathcal{C}_r$ theory, and, moreover adding (9) and (10) of Definition 5.4.1 results the same equations as in $\mathcal{T}$. Then if $\mathcal{T}^-$ is reduction-closed then $\mathcal{T}$ is reduction-closed.

Proof. The proof is very similar to [46].

We seek to characterise insensitive terms as processes which never interact with the surrounding environment. We need to work quite hard for this. We define a notion of standard form similarly to the definition of Milner [58] for the π-calculus. We shall see that this notion of standard form helps us to reason about insensitive processes.

DEFINITION 5.4.9 A process is said to be in *standard form*

$$(\nu p_1 \ldots p_n)(\mathbf{0} \mid M.R_1 \mid \ldots \mid M.R_m \mid (y_1).T_1 \mid \ldots \mid (y_k).T_k \mid \langle q_1\rangle \mid \ldots \mid \langle q_z\rangle \mid n_1\,[P_1] \mid \ldots \mid n_r\,[P_r] \mid !Q_1 \mid \ldots \mid !Q_k)$$

where for h such that $1 \leq h \leq k$ we have that Q_h is itself in standard form.

PROPOSITION 5.4.10 Every process P is structurally congruent to a process in standard form.

Proof. Every restriction (νp) not guarded by a capability, or an input, which is inside an ambient or inside the composition, can be brought outside by the means of the rules $(\nu n)(P \mid Q) \equiv_a P \mid$

$(\nu n)Q$ and $(\nu m)n\,[P] \equiv_a n\,[(\nu m)P]$ of structural congruence. This leaves zero or more terms that are guarded by capability or input, ambients, or replication. Each term of the replication can be treated similarly.

PROPOSITION 5.4.11 Let $an(P) = \emptyset$, then for some j, k, s:

$$P \equiv_a (\nu p_1 \dots p_n)(\mathbf{0} \mid M_1.R_1 \mid \dots \mid M_m.R_m \mid n_1\,[P_1] \mid \dots \mid n_r\,[P_r] \mid !Q_1 \mid \dots \mid !Q_k)$$

where

- $\{n_1 \dots n_r\} \subseteq \{p_1 \dots p_n\}$;
- $(\bigcup_{i=1}^{r} an(P_i) - \{\star\}) \subseteq \{p_1 \dots p_n\}$;
- $\bigcup_{i=1}^{m} an(M_i.R_i) \subseteq \{p_1 \dots p_n\}$;
- $1 \leq i \leq k,\ an(Q_i) = \emptyset$

Proof. By Proposition 5.4.10 every process P has a standard form, thus we enforce that active names are either restricted, or inside an ambient for the case of communication primitives.

Notice that in the standard from written above, the Q_i cannot have communication primitives at the top level.

NOTATION 5.4.12 We write $\mathcal{C}_r \longrightarrow\!\!\!\!\!\rightarrow_a \mathcal{C}'_r$ if and only $\mathcal{C}_r[P] \longrightarrow\!\!\!\!\!\rightarrow_a \mathcal{C}'_r[P]$.

PROPOSITION 5.4.13 Let U be a process such that $U \in \mathsf{Ins}$ and let $\mathcal{C}_r[\,]$ be a reduction context. If $\mathcal{C}_r[U] \longrightarrow_a R$, then either $\mathcal{C}_r \longrightarrow_a \mathcal{C}'_r$ and $R \equiv_a \mathcal{C}'_r[U]$ or $U \longrightarrow_a U'$ and $R \equiv_a \mathcal{C}_r[U']$

Proof. For all $U \in \mathsf{Ins}$ we have $an(U) = \emptyset$. Then by Proposition 5.4.11 U is structurally congruent to a syntactic form such that:

- The name of every top-level ambient is restricted, and every name of top level unguarded capability is restricted; thus the rule RED OPEN cannot be used in conjunction with the context $\mathcal{C}_r[\,]$;
- Every name of a top-level unguarded capability inside the ambient is restricted; thus the rules RED IN and RED OUT cannot be used in conjunction with the context $\mathcal{C}_r[\,]$;
- There are no top-level communication primitives; thus the rule RED COMM cannot be used in conjunction with the context $\mathcal{C}_r[\,]$;
- If $\mathcal{C}_r[U] \longrightarrow_a R$ by RED PAR, then $\mathcal{C}_r[U] \equiv_a P \mid U$ and $\mathcal{C}_r[] = P \mid []$. Then either $P \longrightarrow_a P'$ and $R \equiv_a P' \mid U$ or $U \longrightarrow_a U'$ and $R \equiv_a P \mid U'$;

- If $\mathcal{C}_r[U] \longrightarrow_a R$ by RED AMB, then $\mathcal{C}_r[U] \equiv_a r\ [\mathcal{C}'_r[U]]$ and $\mathcal{C}_r = r\ [\mathcal{C}'_r[\ ]]$ and $\mathcal{C}'_r[U] \longrightarrow_a M$. By induction hypothesis, if $\mathcal{C}'_r[U] \longrightarrow_a M$ then either $M \equiv_a \mathcal{C}_r[U']$ with $U \longrightarrow_a U'$ or $M \equiv_a \mathcal{C}''_r[U]$ with $\mathcal{C}'_r[] \longrightarrow_a \mathcal{C}''_r$. In both cases the proposition is proved.
- If $\mathcal{C}_r[U] \longrightarrow_a R$ by RED RESTR, this case is similar to the previous one.
- If if $\mathcal{C}_r[U] \longrightarrow_a R$ RED PAR, then observe that the proposition follows by induction and Proposition 5.2.9.

The following two lemmas tell us that insensitive processes never interact with the surrounding environment.

LEMMA 5.4.14 Let U be an insensitive process i.e. $U \in \mathsf{Ins}$ and let $\mathcal{C}_r[\ ]$ be a generic context. If $\mathcal{C}_r[U] \longrightarrow\!\!\!\!\!\rightarrow_a R$ then either $R \equiv_a \mathcal{C}_r[U']$ where $U \longrightarrow\!\!\!\!\!\rightarrow_a U'$ and $U' \in \mathsf{Ins}$ or $R \equiv_a \mathcal{C}'_r[U]$ where $\mathcal{C}_r \longrightarrow\!\!\!\!\!\rightarrow_a \mathcal{C}'_r$.

Proof. By induction the number of steps and Proposition 5.4.13.

As usual, a theory is *consistent* if not all processes are identified in the theory.

DEFINITION 5.4.15 An ambient theory is *consistent* if not all processes are identified in the theory.

Finally we can introduce the central notion in this framework: *sound theories.*

DEFINITION 5.4.16 A theory is *sound* if:

- it contains structural congruence;
- it is consistent;
- it is reduction-closed;
- it identifies all the insensitive terms.

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

DEFINITION 5.4.17 Let $\mathcal{T}_{Ins}$ be the theory generated by $\equiv_a$ and identifying all the insensitive processes.

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

DEFINITION 5.4.18 An equational theory $\mathcal{T}$ preserves *strong (weak) barbs* if whenever $\mathcal{T} \vdash P = Q$, then for every name n, if $P \downarrow n$ then $Q \downarrow n$ (if $P \downarrow n$ then $Q \Downarrow n$).

LEMMA 5.4.19 $\equiv_a$ preserves strong barbs.

Proof. By induction on $\equiv_a$.

LEMMA 5.4.20 $\mathcal{T}_{Ins}$ preserves strong barbs.

Proof. By induction on Definition 5.4.1.

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

PROPOSITION 5.4.21 $\mathcal{T}_{Ins}$ is sound.

Proof. *Reduction Closure* : The case of $\equiv_a$ follows from the definition of $\longrightarrow_a$. For the case of the laws identifying insensitive terms, suppose that $\mathcal{C}_r[U] = \mathcal{C}_r[V]$. Then by Lemma 5.4.14 we have that if $\mathcal{C}_r[U]$ reacts (in one step) it is either $\mathcal{C}_r$ that reacts in which case it can be imitated by the context in $\mathcal{C}_r[V]$; or it is U which reacts, $U \longrightarrow_a U'$, and U' is still insensitive in which case $\mathcal{C}_r[V] \longrightarrow\!\!\!\!\!\rightarrow_a \mathcal{C}_r[V]$. By Lemma 5.4.8 the reduction closure property is preserved by all contexts.

Insensitive terms : Included by construction.

Structural Congruence : Included by construction.

Consistency : $n\,[\,] \neq \mathbf{0}$ by Lemma 5.4.20.

So far we have proved that there is at least one sound theory, namely $\mathcal{T}_{Ins}$, which is the *minimal* sound theory according to our definition. A sound theory in general may be more generous, in the sense that it may equate more processes; therefore it is not trivial or automatic that *any* sound theory $\mathcal{T}$ preserves weak barbs. Indeed this is the core of this framework, whose proof relies heavily on the fact that a sound theory is consistent, unlike the proof of Proposition 5.4.21, where consistency was a consequence of the preservation of the barbs. We need the following lemma, which holds for any sound theory.

LEMMA 5.4.22 (*Incompatible pairs [46]*) Let $\mathcal{T}$ be a sound theory. Then, $n\,[\,] \neq \mathbf{0}$, for all n.

Proof. Assume by contradiction that, for some m, $m\,[\,] = \mathbf{0}$. First, we show that for all n, $n\,[\,] = \mathbf{0}$. Now $(\nu m)(\mathsf{open}\, m.n\,[\,] \mid m\,[\,]) = (\nu m)(\mathsf{open}\, m.n\,[\,] \mid \mathbf{0})$. Then:

$$(\nu m)(\mathsf{open}\, m.n\,[\,] \mid m\,[\,]) \longrightarrow_a n\,[\,]$$

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

Now we are going to prove inconsistency. Take any process P, and take m such that $m \notin \mathit{fn}(P)$. We know from above that $m\,[\,] = \mathbf{0}$. So

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

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

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

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

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

PROPOSITION 5.4.23 Let $\mathcal{T}$ be a sound theory. Then it preserves weak barbs.

Proof. Assume $\mathcal{T} \vdash P = Q$ and $P \downarrow n$. Let $A = \mathit{fn}(P) \cup \mathit{fn}(Q)$. Take $m \notin A$. $P \mid \mathsf{open}\, n.m\,[\,] \longrightarrow_a P' \mid m\,[\,]$ and by reduction closure $Q \mid \mathsf{open}\, n.m\,[\,] \longrightarrow\!\!\!\!\!\rightarrow_a Q'$ and $P' \mid m\,[\,] = Q'$. Since $\mathcal{T}$ is a congruence, we conclude $(\nu A)(P' \mid m\,[\,]) = (\nu A)(Q')$. We analyse Q'. If $\mathsf{open}\, n.m\,[\,]$ is no longer a subterm of Q' then immediately $Q \Downarrow n$. Otherwise $\mathsf{open}\, n.m\,[\,]$ is still inside Q' and

$$(\nu A)(Q') = (\nu A)(P' \mid m\,[\,]).$$

Since $(\nu A)(Q')$ is insensitive, we can derive $m\,[\mathbf{0}] = \mathbf{0}$. Contradiction. So $\mathsf{open}\, n.\mathbf{0}$ was consumed after all, proving that $Q \Downarrow n$.

LEMMA 5.4.24 Let $\mathcal{T}$ be a sound theory and $\mathcal{T} \vdash P = Q$. If $P \Downarrow n$ then $Q \Downarrow n$.

Proof. Assume $P \Downarrow n$. This means by definition that $P \longrightarrow\!\!\!\!\!\rightarrow_a P'$ and $P' \downarrow n$. By reduction closure we have that $Q \longrightarrow\!\!\!\!\!\rightarrow_a Q'$ and $P' = Q'$. By Theorem 5.4.23 we have that $Q' \Downarrow n$. Therefore we have that $Q \Downarrow n$.

LEMMA 5.4.25 *i*) Let $\mathcal{T}$ be sound. Then $\mathcal{T} \not\vdash n\,[s\,[\,]] = n\,[\,] \mid s\,[\,]$ for all $n \neq s$.

ii) Let $\mathcal{T}$ be sound. Then $\mathcal{T} \not\vdash n\,[n\,[\,]] = n\,[\,] \mid n\,[\,]$ for all n

iii) Let $\mathcal{T}$ be sound. Then $\mathcal{T} \not\vdash \mathsf{in}\,n = \mathsf{in}\,m$ for all $n \neq m$.

iv) Let $\mathcal{T}$ be sound. Then $\mathcal{T} \not\vdash \mathsf{out}\,n = \mathsf{out}\,m$ for all $n \neq m$.$\mathcal{U}$

Proof. We prove only (i); the other cases are similar. Consider the context $\mathcal{C} = q\,[\mathsf{in}\,n.\mathsf{in}\,s.w\,[\mathsf{out}\,s.\mathsf{out}\,n]]$. Assume that $\mathcal{T} \vdash n\,[s\,[\,]] = n\,[\,] \mid s\,[\,]$; then $\mathcal{C}[n\,[s\,[\,]]] = \mathcal{C}[n\,[\,] \mid s\,[\,]]$. Then $\mathcal{C}[n\,[s\,[\,]]] \Downarrow w$ and $\mathcal{C}[n\,[\,] \mid s\,[\,]] \not\Downarrow w$, which proves our lemma.

NOTATION 5.4.26 If $\mathcal{T}_1$ and $\mathcal{T}_2$ are two theories, we write $\mathcal{T}_1 + \mathcal{T}_2$ to means the union of the the two theories. For a finite index J, we write $\sum_{i \in J} \mathcal{T}$ for the generalized union.

We write $\mathcal{U}$ for the sum of all sound theories.

LEMMA 5.4.27 (*Chain Lemma*) [46] Let $\mathcal{V}$ be $\sum\{\mathcal{T}_j\}_{j \in J}$, a sum of ambient theories. Then, if $\mathcal{V} \vdash P = Q$, we have a chain of equations:

$$\mathcal{T}_{j_0} \vdash P = R_0 \ldots \mathcal{T}_{j_k} \vdash R_{k-1} = R_k \ldots \mathcal{T}_{j_n} \vdash R_{n-1} = Q$$

for some $j_i \in J$, $0 \leq i \leq n$.

Proof. By induction on Definition 5.4.1.

PROPOSITION 5.4.28 Let $\mathcal{T}_j$ be a reduction-closed theory for all $j \in J$. Then $\sum\{\mathcal{T}_j\}_{j \in J}$ is also reduction-closed.

Proof. By (Chain lemma) 5.4.27 we have that there exists n such that

$$\mathcal{T}_{j_0} \vdash P = R_0 \ldots \mathcal{T}_{j_k} \vdash R_{k-1} = R_k \ldots \mathcal{T}_{j_n} \vdash R_{n-1} = Q$$

$\mathcal{T}_{j_0}$ is reduction-closed. Therefore $P \longrightarrow\!\!\!\!\!\rightarrow_a P'$ and $R_0 \longrightarrow\!\!\!\!\!\rightarrow_a R'_0$ and $P' = R'_0$. But also all the other theories are reduction-closed; therefore for all k such that $1 \leq k \leq n$ we have that $R_k \longrightarrow\!\!\!\!\!\rightarrow_a R'_k$ and $R'_{k-1} = R'_k$ and $R'_{n-1} = Q'$. By transitivity we have that $P' = Q'$.

THEOREM 5.4.29 Let $\mathcal{T}_j$ be a sound theory for all $j \in J$. If $\sum\{\mathcal{T}_j\}_{j \in J} \vdash P = Q$ then if $P \downarrow n$ then $Q \Downarrow n$.

Proof. By (Chain lemma) 5.4.27 we have that there exists n such that

$$\mathcal{T}_{j_0} \vdash P = R_0 \ldots \mathcal{T}_{j_k} \vdash R_{k-1} = R_k \ldots \mathcal{T}_{j_n} \vdash R_{n-1} = Q$$

Since $\mathcal{T}_{j_0}$ is sound, then if $P \downarrow n$ then $R_0 \Downarrow n$. For all $0 < j < n$ we have that $R_j \Downarrow n$ implies $R_{j+1} \Downarrow n$ and if $R_{n-1} \Downarrow n$ then $Q \Downarrow n$. By transitivity we have that $Q \Downarrow n$.

THEOREM 5.4.30 Let $\mathcal{U}$ be the union of all sound theories. Then $\mathcal{U}$ is sound.

Proof. By Proposition 5.4.28 $\mathcal{U}$ is reduction-closed, and clearly contains structural congruence, $\equiv_a \subset \mathcal{U}$ and the insensitive terms $\mathsf{Ins} \subset \mathcal{U}$. By Proposition 5.4.29, if $\mathcal{U} \vdash P = Q$ and $P \Downarrow n$ then $Q \Downarrow n$. Thus $\mathcal{U} \nvdash n\,[\,] = \mathbf{0}$. Therefore $\mathcal{U}$ is consistent.

5.4.1 *The importance of the insensitive terms*

In the previous section we have shown that new sound theories can be built up by means of the union operation. In particular, the main result concerns the possibility of constructing a maximal sound theory, as the union of all sound theories. The previous result, however, is not trivial, and insensitive terms play crucial role. In this section we are going to show that consistent and reduction-closed theories do not admit a maximal theory. Thus, insensitive terms are essential in the previous construction.

DEFINITION 5.4.31 We define $\mathcal{F}_1$ as containing the following equations:

(0)	$(\nu n)\mathsf{open}\, n \mid P = (\nu n)\mathsf{open}\, n \mid Q$	(5)	$P = Q \Longrightarrow M.P = M.Q$
(1)	$P = Q$ if $P \equiv_a Q$	(6)	$P = Q \Longrightarrow (\nu w)P = (\nu w)Q$
(2)	$P = Q \Longrightarrow Q = P$	(7)	$P = Q \Longrightarrow n\,[P] = n\,[Q]$
(3)	$P = Q \;\; Q = Z \Longrightarrow P = Z$	(8)	$P = Q \Longrightarrow (y).P = (y).Q$
(4)	$P = Q \Longrightarrow P \mid R = Q \mid R$	(9)	$P = Q \Longrightarrow \sigma(P) = \sigma(Q)$
$(4')$	$P = Q \Longrightarrow R \mid P = R \mid Q$	(10)	$P = Q \Longrightarrow !P = !Q$

DEFINITION 5.4.32 We define $\mathcal{F}_2$ as containing the following equations:

(0)	$(\nu n)\mathsf{open}\, n = \mathbf{0}$	(5)	$P = Q \Longrightarrow M.P = M.Q$
(1)	$P = Q$ if $P \equiv_a Q$	(6)	$P = Q \Longrightarrow (\nu w)P = (\nu w)Q$
(2)	$P = Q \Longrightarrow Q = P$	(7)	$P = Q \Longrightarrow n\,[P] = n\,[Q]$
(3)	$P = Q \;\; Q = Z \Longrightarrow P = Z$	(8)	$P = Q \Longrightarrow (y).P = (y).Q$
(4)	$P = Q \Longrightarrow P \mid R = Q \mid R$	(9)	$P = Q \Longrightarrow \sigma(P) = \sigma(Q)$
$(4')$	$P = Q \Longrightarrow R \mid P = R \mid Q$	(10)	$P = Q \Longrightarrow !P = !Q$

PROPOSITION 5.4.33 The theories $\mathcal{F}_1$ and $\mathcal{F}_2$ are reduction-closed and consistent.

Proof. We show that $\mathcal{F}_1$ is (i) reduction-closed and (ii) consistent.

By induction on the clauses of Definition 5.4.31, if $\mathcal{F}_1 \vdash P = Q$ then either $P \equiv_a Q$ or for some terms P', Q' and contexts $\mathcal{C}_1, \mathcal{C}_2$ $P = \mathcal{C}_1[(\nu n)\mathsf{open}\, n \mid P']$ and $Q = \mathcal{C}_2[(\nu n)\mathsf{open}\, n \mid Q']$.

i) Assume that $\mathcal{F}_1 \vdash P = Q$. Then for some term P' and context $\mathcal{C}$ we have $P \equiv_a \mathcal{C}[(\nu n)\mathsf{open}\, n \mid P']$. Thus $P \longrightarrow_a R'$ then either $P \longrightarrow_a \mathcal{C}[(\nu n)\mathsf{open}\, n \mid P'']$ or $P \longrightarrow_a \mathcal{C}'[(\nu n)\mathsf{open}\, n \mid P']$. Notice that in both cases $(\nu n)\mathsf{open}\, n$ will never disappear. Now for some term Q' and context $\mathcal{C}^*$ $Q \equiv_a \mathcal{C}^*[(\nu n)\mathsf{open}\, n \mid Q']$. Thus, $Q \longrightarrow\!\!\!\!\rightarrow_a Q$ and the reduction closure is proved.

ii) Consistency follows by observing that the terms $P = \mathsf{in}\, n.\mathbf{0}$ and $Q = \mathsf{out}\, n.\mathbf{0}$ are neither structurally congruent $P \not\equiv_a Q$ nor contain the term $(\nu n)\mathsf{open}\, n$, i.e. $P \not\equiv_a \mathcal{C}_1[(\nu n)\mathsf{open}\, n \mid P']$ and $Q \not\equiv_a \mathcal{C}_2[(\nu n)\mathsf{open}\, n \mid Q']$.

Consistency and reduction closure for $\mathcal{F}_2$ can be proved in a similar way.

THEOREM 5.4.34 $\mathcal{F}_1 + \mathcal{F}_2$ is inconsistent.

Proof.
$$\begin{aligned} \mathcal{F}_1 + \mathcal{F}_2 &\vdash (P \mid (\nu n)(\mathsf{open}\, n) = (Q \mid (\nu n)(\mathsf{open}\, n) \\ \mathcal{F}_1 + \mathcal{F}_2 &\vdash P \mid \mathbf{0} = Q \mid \mathbf{0} \\ \mathcal{F}_1 + \mathcal{F}_2 &\vdash P = Q \end{aligned}$$

COROLLARY 5.4.35 (*cf.[46]*) There exists no maximal, consistent and reduction-closed theory.

Proof. Clearly by Theorem 5.4.34.

The final result shows that there is an operational characterisation of a maximal sound theory, namely contextual barbed bisimulation.

THEOREM 5.4.36 $\approx_a = \mathcal{U}$.

Proof. First we show that $\mathcal{U}$ is a contextual barbed bisimulation, which establishes $\mathcal{U} \subseteq \approx_a$. We show that

$$\mathcal{S} = \{(P, Q) : \mathcal{U} \vdash P = Q\}$$

is a contextual barbed bisimulation. The symmetry of $\mathcal{S}$ is derived by the symmetry of $\mathcal{U}$. By Proposition 5.4.23 it holds that if $P \downarrow n$ then $Q \Downarrow n$. For all contexts $\mathcal{C}$, by definition of sound theory, $\mathcal{U} \vdash \mathcal{C}[P] = \mathcal{C}[Q]$. If $\mathcal{C}[P] \longrightarrow_a P'$, then by reduction closure of $\mathcal{U}$, for some Q' we have that $\mathcal{C}[Q] \longrightarrow\!\!\!\!\rightarrow_a Q'$ with $\mathcal{U} \vdash P' = Q'$.

Conversely we must prove that $\approx_a$ is a sound theory. It is reduction-closed by definition and consistent since $n\,[\,] \not\approx_a \mathbf{0}$ and it contains structural congruence $\equiv_a \subseteq \approx_a$. Moreover, for all

P, Q, if $P \approx_a Q$ then $\mathcal{C}[P] \approx_a \mathcal{C}[Q]$ since $\approx_a$ is a congruence. It identifies the insensitive terms, since

$$\mathcal{S} = \{(\mathcal{C}[P], \mathcal{C}[\mathbf{0}]) : P \in \mathsf{Ins}\} \cup \{(\mathcal{C}[\mathbf{0}], \mathcal{C}[P]) : P \in \mathsf{Ins}\}$$

is a contextual barbed bisimulation.

The family of sound theories can be then seen as a lattice; we have characterised the minimal sound theory $\mathcal{T}_{Ins}$ and the maximal sound theory $\mathcal{U}$. We can say very little about the sound theories that lie in between. Certainly, structural congruence is a contextual bimisulation, yet not a sound theory, since it does not contain insensitive terms.

5.5 *Concluding remarks*

We have shown that the Honda and Yoshida construction is general enough to be adapted to MA. One interesting and important point to be made is that the same construction can be adapted to other dialects such as PAC and SA as shown in [73, 87]. The main difference between the construction carried out for MA, PAC and SA lies in the choice of insensitive names. For PAC the critical point is the $\mathsf{push}\, n.P$, which requires that the name $\star$ is added to the definition of active names, i.e. $an(\mathsf{push}\, n.P) = \{\star, n\}$. This is necessary for the pathological behaviour of the term $P = (\nu n)(\mathsf{push}\, n \mid n\,[\,])$. If we eliminate the $\star$ from the definition of active names and consider the process P, we can easily see that it might react the context $w\,[\,]$, and in doing so, it would invalidate Lemma 5.4.14.

As regards SA, the definition of active names is sophisticated in order to take into account the co-actions. The basic construction for the maximal sound theory remains the same.

Chapter 6

Expressiveness results

In this chapter we shall compare the expressiveness of ambient calculi against different dialects of the π-calculus. To this end we consider a standard problem in the literature of distributed algorithms [50, 4, 80]: *leader election problem in symmetric networks.* We review the literature on this in the first section.

The asynchronous π-calculus [45, 10] has been encoded into MA [24] with the use of the communication primitives. Moreover there has been an encoding of the asynchronous π-calculus in two dialects: the Push and Pull Ambient Calculus [73] and Safe Ambients [49]. The synchronous π-calculus without choice has been encoded into SA [49] without communication [92]. These encodings show that, in the ambient world, the behaviour of the asynchronous π-calculus can be simulated. This seems to imply that MA is at least as expressive as the π-calculus (without choice). Of course, this poses the question whether the ambient calculus (or any of its dialects) is more (or equally) expressive with respect to any of the dialects of the π-calculus. Palamidessi [71] adapts the *leader election problem* to the labelled transition semantics of the π-calculus, and shows that the π-calculus with mixed choice cannot be encoded (under certain conditions) in the π-calculus with separate choice. Taking inspiration from her work, we adapt the leader election problem to the general framework of reduction semantics. In this way we are able to compare under *identical conditions* different calculi. We restate Palamidessi's results for the π-calculus, and we show that the pure version of MA without restriction and without the open capability admits symmetric electoral systems, that is, it is possible to solve the problem of electing a leader in a symmetric network. By the work of Palamidessi, this implies that this fragments of the ambient calculus is not encodable (under certain conditions) in the π-calculus with separate choice. Moreover, we discuss which fragment of other dialects of the ambient calculus (PAC and SA) admit an electoral

system. We shall present a hierarchy within MA, by showing that MA without the in capability *does not* admit an electoral system. In the conclusions we discuss the implementation of MA into the Distributed Join Calculus [34].

6.1 *Leader election problems*

Expressiveness results have positive aspects, where encodings are given; for instance we have seen the encoding of the matching operator (Chapter 3) and of the asynchronous π-calculus (Chapter 4 §4.4). However, one can also show that some encodings do not exist. One way of proceeding is to show that there is a problem that can be solved in one calculus, but not in the other. This method is used in the field of distributed algorithms [50, 80]. Our predecessors [12, 70, 28, 71] have applied these techniques to process calculi with labelled transition system semantics. We seek to adapt these techniques to calculi with reduction semantics (Chapter 1).

In the field of distributed algorithms [50, 80], various models of computation are compared via *leader election problems.* We talk of problems in the plural, because there are different settings that lead to diverse solutions (when solutions do exist). The *leader election problem* is solved, if starting from a configuration of processes in the *same state* any possible computation reaches a configuration where *one* process is in the state of *leader* and the other processes are in the state *lost* (i.e. they have lost the election). It is important to notice that there *cannot* be more than one leader, and the leader has to be one of the processes in the configuration. We borrow from Tel [80] the words ‘configuration’ and ‘being in the same state’ in order to describe the problem in its most general terms. For ‘configuration’ one can think of a number of processes running in parallel; ‘being in the same state’ means that from the computational point of view, all process behave in the same way. Tel clearly define leader election problems [80] (Definition 7.1). Here we summarise and discuss the crucial criteria:

Symmetry : Each process in the configuration has to have the same duties.

> This is a necessary requirement in order not to trivialise the problem, in the sense that the solution would become too easy. In fact, in an asymmetric configuration of processes one process can declare itself the winner. Notice that, this is not possible in symmetric configuration, since if one can declare itself the winner, every other process in the configuration can do the same. Thus, in symmetric networks, for the winner to be elected, the *initial symmetry* has to be somehow broken.

Distribution : The computation has to be *decentralised*, in the sense that the computation has to start from any subset of processes in the network or configuration.

Again, this is a crucial requirement. In general, as Tel [80] points out, leader election problems are run after a crash of a system, to the end of establishing which process can start the initialisation. In this context, the configuration of processes has to be able to elect a leader without any help from outside. Moreover, without this requirement, the problem would be again trivialise, since the presence of an external process outside the configuration would compromise the requirement on symmetry.

Uniqueness of the leader : The set of network reaches a *terminal configuration* from *any* computation. In the terminal configuration there is *one process only* that is elected the *winner* and the other processes in the configuration have lost.

This requirement specifies the successful election. The election of the leader in this setting does not happen 'democratically', in the sense that every single process in the network has to agree on the winner. In the presence of more winners, one of which might have the majority of 'votes', or alternatively in the presence of no winners, the election fails, i.e. there is no solution. In leader election problems a solution exists, only if any possible computation elects one winner only. It does not matter if in a given network, there might be different computations that elect different winners.

We will precisely specify these three requirements in the next section, where we will present the general framework in which the leader election problem is defined.

Leader election problems raise different solutions by modifying the following parameters:

TOPOLOGY OF THE NETWORK : The network could be a *fully connected graph* or a *ring* or *tree* or any other graph or hyper-graph [3, 80, 50].

SIZE OF THE NETWORK : The number of processes can be known or unknown before starting the election [80].

DECLARATION OF THE LEADER : The leader or *winner of the election* could be announced by one process only, that could be the leader itself or any other process. Alternatively every process in the configuration has to be aware of the winner. The latter requirement is considered standard, although the weaker one (the former one) is also acceptable, since the winner could inform the other processes of the outcome of the election.

Finally, in distributed systems different algorithms for electing a leader, differentiate models on complexity issues. Bougé [12] called these *qualitative approaches*. We do not take complexity issues into account here, since these issues are independent from languages. For instance, as Bougé points out, lower bound results are strictly related to the knowledge of the size of the network, or whether the identifiers of the processes are integers and so on. These kind of issues are

not relevant in our setting.

We have described the leader election problem as presented in the field of distributed algorithms. In this field, it is common to reason on what Nestmann [66] calls *pseudo-code.* This means that proofs are given by using some form of 'general-enough-language'[1], that is without a formalised semantics. Nestmann shows that this approach very often hides underpinning problems and assumptions. He proves this for *consensus problems* and he defines an appropriate process calculus. It is fair to conclude that it is a big step to translate informal reasoning on leader election problems [50, 80] to a process calculus with a formal semantics. The first significant attempt in this field was made by Bougé [12]. He formalised the notion of leader election problem in symmetric networks for CSP [44]. He defines for CSP the notion of *symmetric network* as a composition of processes (as we intend with the operator $|$), such "*that all the processes exhibit equivalent behaviour under suitable conditions*"[2]. Bougé defines also the notion of *symmetric electoral system*: as the existence of a solution to electing a winner in symmetric networks. The most remarkable achievements are the separation results between CSP with input and output guards and CSP with input guards only (Theorem 5.1.1) and between the latter and CSP without guards (Theorem 5.1.2) based on the notion of *symmetric reasonable implementation.* Bougé does not prove his theorems by using the formal semantics of CSP; he still works in the pseudo-code framework. A decisive step towards formalization of the notion of leader election problem was made by Palamidessi [70, 71] for the π-calculus. This work has been the major source of inspiration for this chapter and [74]. Palamidessi clearly formalised the notion of *symmetric electoral system* and clarified the necessary requirements on the encoding for her negative results. On the latter, unlike Bougé [12], she introduces the notion of *reasonable semantics*, which "should distinguish processes which differ on the observables of their maximal computations". Palamidessi proves *formally* (i.e by the labelled transition system) for the first time (Theorem 4.2), that any symmetric network in π-calculus with separate choice admits a computation that never breaks the initial symmetry. This result is used to show that there is no encoding of the π-calculus with mixed choice into the π-calculus with separate choice ([71] Corollary 7.1). In her paper Palamidessi still uses a graph framework, as in the tradition of distributed algorithms [50, 80, 3, 12], and she proves that CCS [54] does not admit a symmetric electoral system in a ring, as opposed to the π-calculus with mixed choice [71] (Corollary 7.2). Using a similar approach to Palamidessi, Ene and Muntian [28] show that the π-calculus with broadcasting primitives cannot be encoded in the standard π-calculus. Our predecessors [28, 71] and in some sense also Bougé [12] worked in

[1] This definition is ours.

[2] The italics are ours.

the tradition of labelled transition systems. We believe that reduction semantics is an appropriate formalisation for the ambient world, since there is no agreement within the scientific community about what constitutes a good labelled transition system. Moreover it is a unifying framework that allows us compare different calculi under the same conditions.

6.2 *A general framework for reduction semantics*

In this section we define networks and electoral systems for calculi equipped with reduction semantics (Section 1.2). Hence the definitions below apply equally well to the ambient calculus and to the π-calculus, and to any other calculus that uses the reduction semantics framework as described in Chapter 1. This is the contribution we bring to the research. In this section, we shall present definitions of: *symmetric network*, *computation*, and *electoral system*, in order to define the leader election problem. A *symmetric network* is a composition of processes that differ only up to the injective renaming of processes. This is a way of specifying in this setting that each process has the same duties. This notion with minor changes is inherited from the literature [71]. A *computation* is the way in which we talk about the evolution of the program. In this case, our definition of computation is different from Palamidessi's [71]. She uses labelled transition systems, while we use reduction relations (i.e. τ-actions). The *electoral system* defines the strategy for electing leader. We differ from [71] in two ways: first of all, the winner is elected if the name of the winner can be detected as a barb. Palamidessi [71] uses actions from the label transition system $\overline{out}\langle i\rangle$. Moreover, for Palamidessi the requirement for an electoral system is that *every* process in the electoral system can execute a special action $\overline{out}\langle i\rangle$. In other words everyone is aware of the leader. Our notion is weaker, in that we merely require that *at least one* process announces the winner, and it is left open how many processes make the announcement. We do not distinguish processes that are in *the state of loss*, thus we differ from the approach in distributed systems described above. We shall see that the lost-state of a process can be represented, but it is not essential for the election (Remark 6.2.20).

NOTATION 6.2.1 For the purpose of this chapter, we assume that the set of names $\mathcal{N}$ includes a set of *observables*:

$$\mathsf{Obs} = \{\omega_i : i \in \mathsf{IN}\}$$

such that for all i, j, $\omega_i \neq \omega_j$ if $i \neq j$. The observables will be used by networks to communicate with the outside world, and *can never be restricted* (i.e. they never occur under the scope of restriction). We also assume that the names contain a special subset that is the natural numbers:

$$\mathsf{IN} \subset \mathcal{N} \quad \text{and} \quad \mathsf{IN} \cap \mathsf{Obs} = \emptyset.$$

As Palamidessi points out [70, 71] this is simply notation; there is no need to add the natural numbers as datatype. We also use *natural numbers* as indices of processes in a network, for instance P_i where P is a process and i is a natural number.

REMARK 6.2.2 We remind the reader that in the reduction semantics framework we assume that a generic process calculus is equipped with a set of names $\mathcal{N}$, reduction relation $\longrightarrow$ and barb $P \downarrow n$. Moreover, in the following section we assume that the calculus has restriction (ν) and parallel composition operators ($|$), and the notion of structural congruence $\equiv$.

Networks are just collections of processes running in parallel, possibly equipped with some global variables.

DEFINITION 6.2.3 (*Network*) A *network* N of size k is a process in the form

$$(\nu x_0, \ldots, x_{l-1})(P_0 \mid P_1 \mid P_2 \mid \ldots \mid P_{k-1})$$

We will use the notation $[P_0 \mid P_1 \mid \ldots \mid P_{k-1}]$ for representing the network above when the globally bound variables $x_0, \ldots, x_{l-1}$ are not relevant.

DEFINITION 6.2.4 (*Permutation on names*)

i) A *permutation on names* is a bijection $\sigma : \mathcal{N} \to \mathcal{N}$ such that σ is a bijection both on Obs and on $\mathbb{N}$, and σ is consistent between Obs and $\mathbb{N}$, in the sense that for all $i \in \mathbb{N}$, $\sigma(\omega_i) = \omega_{\sigma(i)}$.

ii) A permutation σ over a finite set E of natural numbers is a permutation on names such that σ restricted to E is a bijection.

REMARK 6.2.5 Any permutation σ gives rise in a standard way to a mapping on processes, where $\sigma(P)$ is the same as P, except that any free name n of P is changed to $\sigma(n)$ in $\sigma(P)$, with bound names being adjusted as necessary to avoid clashes. In other words, the definition of $\sigma(P)$ where σ is a permutation, is recursively defined on the syntax of the language by avoiding that names are captured by binding operators. We cannot give a formal definition, without talking of a concrete language; however, the definition of permutation for the π-calculus with mixed choice would follow Definition 2.1.5 and for MA it would follow Definition 2.2.3.

DEFINITION 6.2.6 (*Automorphism in a network*) Let N be a network of size k:

$$\mathsf{N} \stackrel{def}{=} (\nu x_0, \ldots, x_{l-1})(P_0 \mid P_1 \mid \ldots \mid P_{k-1}).$$

A *network automorphism* σ is a permutation over a finite subset of natural numbers $E \stackrel{def}{=} \{0, 1, \ldots, k-1\}$ such that σ preserves the distinction between free and bound names, i.e. $x \in \{x_0, \ldots, x_{l-1}\}$ iff $\sigma(x) \in \{x_0, \ldots, x_{l-1}\}$.

Intuitively a network N is symmetric with respect to an automorphism σ if and only if for each i the renaming of the process associated to i is the same up to alpha-conversion the process associated to the permuted index $\sigma(i)$.

NOTATION 6.2.7 The *composition* of automorphisms $\sigma \bullet \theta$ is defined in the usual manner.

DEFINITION 6.2.8 (*Orbit*) Let N be a network of size k and σ an automorphism on it. For any $i \in \{0, \ldots, k-1\}$ the *single orbit* $\mathcal{O}_\sigma(i)$ generated by σ is defined as follows:

$$\mathcal{O}_\sigma(i) \stackrel{def}{=} \{i, \sigma(i), \sigma^2(i) \ldots \sigma^{k-1}(i)\}$$

where σ^j represents the composition of σ with itself j times.

We can now give the definition of symmetric network. In the first part of the following definition we closely follow Palamidessi [70, 71]. In the second part however, we restrict the notion of symmetry to networks where the automorphism has one orbit only, since this is the only notion that is necessary for proving the result in this thesis.

DEFINITION 6.2.9 (*Symmetric Network*)

i) Let $\mathsf{N} = [P_0 \mid \ldots \mid P_{k-1}]$ be a network and σ an automorphism on it. We say that N is *symmetric with respect to* σ iff for each $i \in \{0, \ldots, k-1\}$, $P_{\sigma(i)} = \sigma(P_i)$ holds.

ii) A network $\mathsf{N} = [P_0 \mid \ldots \mid P_{k-1}]$ is called *symmetric* if it is symmetric with respect to some automorphism with a single orbit.

REMARK 6.2.10 In Chapter 2 and Chapter 4 where we have introduced the π-calculus and the different ambient calculi we have assumed the renaming of bound variable as a sort of *primitive operation*, in the sense that two processes that differ on their bound variables only are considered the same. In general by our definition two processes P, Q are the same $P = Q$, either because they are the same process or because they differ on their bound variables. Canonically in the π-calculus [58, 78], however, terms that differ on their bound variables only are related by a relation called α-*conversion*, sometimes written $\equiv^\alpha$. Palamidessi [70, 71] defines symmetric network by the means of $\equiv^\alpha$, while we can simply use equality $=$ for the reason explained above. In other words, in this context choosing the relation $\equiv^\alpha$ or $=$ does not make *any difference*.

Our definitions of network automorphism and symmetric network differ from those of Palamidessi, since she takes the network topology into account, and associates a hypergraph with a network. Automorphisms are defined with respect to this hypergraph, and a network is symmetric if it

is symmetric with respect to *every* automorphism. As stated above, we have chosen our definitions because they seem to capture exactly what is needed for our separation results.

DEFINITION 6.2.11 Consider a network $\mathsf{N} = [P_0 \mid P_1 \mid \dots P_{k-1}]$. A computation $\mathcal{C}$ is a (finite or infinite) sequence:

$$\mathsf{N} \longrightarrow \mathsf{N}^1 \longrightarrow \mathsf{N}^2 \longrightarrow \cdots \longrightarrow \mathsf{N}^j \longrightarrow \cdots .$$

A computation $\mathcal{C}$ is *maximal* if it cannot be extended, i.e. either $\mathcal{C}$ is infinite, or else it is of the form $\mathsf{N} \longrightarrow \mathsf{N}^1 \longrightarrow \mathsf{N}^2 \cdots \longrightarrow \mathsf{N}^h$ where $\mathsf{N}^h \not\rightarrow$.

Our notion of computation is defined using the reduction relation only. This is a substantial difference from previous authors [70, 71, 28], where computation is defined as a sequence of transitions derived from the labelled transition system. We will see that the separation result for the π-calculus still holds in this setting; however, there is a difference in the use of restriction as we will explain in Section 6.2.1.1. Notice that in the previous definition of computation, we assume that a network reduces to a network. This might be seen as restrictive; however since any process can be seen as a network, the above definition is general enough. Moreover notice that we do not require that the network preserves its own size. In fact in general this might not even be true, as will become clear when presenting electoral systems in the ambient calculus.

DEFINITION 6.2.12 Let $\mathcal{C}$ be a computation $\mathsf{N} \longrightarrow \cdots \longrightarrow \mathsf{N}^h \longrightarrow \cdots$. We define the *observables of* $\mathcal{C}$ as

$$\mathsf{Obs}(\mathcal{C}) = \{\omega \in \mathsf{Obs} : \exists h \ \mathsf{N}^h \downarrow \omega\}.$$

REMARK 6.2.13 In the case of the π-calculus, where both input and output barbs are possible, we only observe output barbs, and the preceding becomes:

$$\mathsf{Obs}(\mathcal{C}) = \{\omega \in \mathsf{Obs} : \ \exists h \, \mathsf{N}^h \downarrow \bar{\omega}\}.$$

Intuitively an electoral system is a network which reports a unique winner no matter how the computation proceeds.

DEFINITION 6.2.14 (*Electoral system*) A network $\mathsf{N} = [P_0 \mid P_1 \mid \dots \mid P_{k-1}]$ is an electoral system if for every maximal computation $\mathcal{C}$ of N there exists an $i < k$ such that $\mathsf{Obs}(\mathcal{C}) = \{\omega_i\}$. An electoral system is said to be *symmetric* if the network is symmetric.

Thus each maximal computation gives exactly one winner. It does not matter which process in the original network displays the observable barb; indeed, in MA this is not even necessarily meaningful, as processes can intermingle using movement capabilities.

Moreover, for Palamidessi the requirement for an electoral system is that *every* process in the electoral system can execute a special action $\overline{out}\langle i\rangle$. In other words everyone is aware of the leader. As she states, her results would hold under the alternative requirement that *exactly one* process announces the winner. Our notion is weaker, in that we merely require that *at least one* process announces the winner, and it is left open how many processes make the announcement.

In the next sections, we are going to show formally that our notion of computation and electoral system do not change her result for the asynchronous π-calculus, no on the π-calculus with mixed choice.

6.2.1 *Calculi with symmetric electoral systems*

In this section we shall present calculi and fragments of them that can solve the leader election problem in symmetric networks. We shall be interested in some fragments of the π-calculus and MA, which we report below.

DEFINITION 6.2.15

i) We define $\pi_m^{-\nu}$ to be the π-calculus with mixed choice but without restriction.

ii) We define MA^{io} to be MA without communication, restriction and open capability.

iii) We define PAC^{pp} to be the PAC without communication, restriction and the open capability.

iv) We define SA^{iop} to be SA without communication, restriction and the out capability.

6.2.1.1 *The π-calculus with mixed choice*

The π-calculus with mixed choice can elect a leader in a symmetric network according to Palamidessi's criteria [71]. It is not difficult to see that π_m admits a symmetric electoral system also according to our new and weaker criteria. In the reduction semantics frameworks, we are able to improve slightly Palamidessi's result; in fact we can show that there exists an electoral system in the π-calculus with mixed choice but without restriction. In the following proposition we show that there exists a symmetric electoral system of size 2 in $\pi_m^{-\nu}$, which is sufficient for restating the separation result in [71] Palamidessi's [71] defines a general way of solving the leader election problem. We believe it is possible in our framework as well; however, it would be outside the scope of this thesis to indulge longer on this matter.

PROPOSITION 6.2.16 In $\pi_m^{-\nu}$ there exists a symmetric electoral system of size 2.

Proof. $$P_0 \stackrel{def}{=} x_0(y) + \overline{x_1}\langle z\rangle.\overline{\omega_0}\langle z\rangle \qquad P_1 \stackrel{def}{=} x_1(y) + \overline{x_0}\langle z\rangle.\overline{\omega_1}\langle z\rangle$$
$$\mathsf{N} \stackrel{def}{=} P_0 \mid P_1$$

The network is symmetric with respect to a single-orbit automorphism σ defined as follows:

$$\sigma(0) = 1 \quad \sigma(1) = 0 \quad \sigma(x_0) = x_1 \quad \sigma(x_1) = x_0 \quad \sigma(\omega_0) = \omega_1 \quad \sigma(\omega_1) = \omega_0$$

with σ the identity on all other names. There are only two possible computations. We present one in details, the other one is identical up to the renaming of σ.

$$\mathcal{C} : \mathsf{N} \longrightarrow {}^{1}\overline{w_1}\langle z \rangle$$

Thus, we conclude $\mathsf{Obs}(\mathcal{C}) = \{\omega_1\}$.

In the previous proposition there are two important features to notice. The first one is that the link-passing capability of the π-calculus plays no rôle; it is the mixed choice which is important. In other words, the previous electoral system could have been written in CCS without link passing. Palamidessi has shown that the π-calculus with mixed choice is not encodable in CCS with value passing. She showed that electoral system exist also in networks that are not fully connected of [71] . In CCS this is not true. The relationship between CCS and the π-calculus is outside the scope of this thesis; however, we claim the same result could be proven in our framework.

The second feature is that the process above would not be an electoral system in Palamidessi's framework. In fact, not every computation leads to the election of a leader, if computation is defined in terms of a labelled transition system. Thus restriction is necessary for Palamidessi, as her Example 4 ([71]) shows.

6.2.1.2 *A small fragment of* MA

In this section we show that there exists an electoral system in MA without communication primitives, restriction, and the open capability. First we show that there exists an electoral system of size 2. The we present a more general solution, for a network of any size. This is one of the main contributions of this dissertation. As far as we know, nobody before has proposed a solution for leader election problem in any of the ambient dialects.

PROPOSITION 6.2.17 In MA^{io} there exists a symmetric electoral system of size 2.

Proof. Let

$$\mathsf{N} \stackrel{def}{=} 0\ [\mathsf{in}\, 1.\omega_0\ [\mathsf{out}\, 0.\mathsf{out}\, 1]] \mid 1\ [\mathsf{in}\, 0.\omega_1\ [\mathsf{out}\, 1.\mathsf{out}\, 0]].$$

The network is symmetric with respect to a single-orbit automorphism σ defined as follows:

$$\sigma(0) = 1 \quad \sigma(1) = 0 \quad \sigma(\omega_0) = \omega_1 \quad \sigma(\omega_1) = \omega_0$$

with σ the identity on all other names.

There are only two possible computations. We shall present the first one in detail:

$$\begin{array}{rl} \mathcal{C}: \quad 0\,[\mathsf{in}\,1.\omega_0\,[\mathsf{out}\,0.\mathsf{out}\,1]] \mid 1\,[\mathsf{in}\,0.\omega_1\,[\mathsf{out}\,1.\mathsf{out}\,0]] & \longrightarrow^{1} \\ 1\,[0\,[\omega_0\,[\mathsf{out}\,0.\mathsf{out}\,1]] \mid \mathsf{in}\,0.\omega_1\,[\mathsf{out}\,1.\mathsf{out}\,0]] & \longrightarrow^{2} \\ 1\,[\omega_0\,[\mathsf{out}\,1] \mid 0\,[\,] \mid \mathsf{in}\,0.\omega_1\,[\mathsf{out}\,1.\mathsf{out}\,0]] & \longrightarrow^{3} \\ \omega_0\,[\,] \mid 1\,[0\,[\,] \mid \mathsf{in}\,0.\omega_1\,[\mathsf{out}\,1.\mathsf{out}\,0]] & \end{array}$$

Thus we conclude $\mathsf{Obs}(\mathcal{C}) = \{\omega_1\}$. The other computation is identical up to renaming via σ.

Before presenting the general solution we have to introduce some notation.

NOTATION 6.2.18 For $0 \leq i < k$, let $\mathbf{S}_i^k \stackrel{def}{=} \{0, \ldots, i-1, i+1, \ldots, k-1\}$, i.e. the natural numbers less than k excluding i. Let $\mathbf{T}_i^k$ be the set of all strings of length $k-1$ using the members of $\mathbf{S}_i^k$ exactly once each. Given an element s of $\mathbf{S}_i^k$ we denote by s^- the string which is s in reverse order. With $\mathsf{in}\,(s)$ we mean the sequence of $\mathsf{in}\,j$ capabilities for each successive $j \in s$. Thus $\omega_2\,[\mathsf{in}\,(s).\mathsf{out}\,(s^-)]$ such that $s \in \mathbf{T}_1^3$ is an abbreviation for the following process:

$$\omega_2\,[\mathsf{in}\,0.\mathsf{in}\,2.\mathsf{out}\,2.\mathsf{out}\,0]$$

THEOREM 6.2.19 In MA^{io}, for any k, there exists a symmetric electoral system of size k.

Proof. The network is defined as:

$$\begin{array}{rcl} P_i & \stackrel{def}{=} & i\,[\prod_{j \in \mathbf{S}_i^k} \mathsf{in}\,j \mid \prod_{s \in \mathbf{T}_i^k} \omega_i\,[\mathsf{in}\,(s).\mathsf{out}\,(s^-).\mathsf{out}\,i]] \\ \mathsf{N} & \stackrel{def}{=} & P_0 \mid \cdots \mid P_{k-1}. \end{array}$$

It is quite easy to show that N is symmetric with respect to the automorphism σ such that $\sigma(i) = i+1$ and $\sigma(\omega_i) = \omega_{i+1}$, if $i \in \{0, 1, 2, \ldots, k-1\}$, and σ is the identity on the rest of the names. The idea is that the processes that take part in the election can enter one another, until they form a linear stack. At this point no further movement of the main ambient is possible, and the leader is the ambient i which is at the top of the stack. For some $s \in \mathbf{T}_i^k$, an ambient ω_i can descend to the bottom of the stack using $\mathsf{in}\,(s)$, and then ascend to the top of the stack using $\mathsf{out}\,(s^-)$. (Of course, it may start this process before the stack is fully formed.) Finally ω_i emerges at the top level using $\mathsf{out}\,i$, and i is declared as the winner. Any ω_j ambient ($j \neq i$) will not be able to use up all its $\mathsf{in}\,(s)$ capabilities, and so will not be able to emerge at the top level. Hence, exactly one winner is declared for each computation. Notice that it is the in capability which breaks symmetry and chooses the winner. The out capability is required to report the winner.

REMARK 6.2.20 In our solution, the symmetric network shrinks to one process only, the winner. One might find this bizarre, because in some sense, other processes in the network have disappeared inside the winner and their identity is lost. This is not problematic for two reasons: there

is no criterion in the definition of the leader election problem that forbids this; moreover, it is possible to modify the previous solution in such a way that losers do not disappear and they might even exhibit a state of loss: a barb ls_i. Below, we present a new solution, observing that this is not significant for the separation results.

NEW SOLUTION FOR LEADER ELECTION PROBLEM

$$P_i \stackrel{def}{=} n_i\,[\mathsf{in}\,\mathsf{ls}_i] \mid i\,[\prod_{j\in\mathbf{S}_i^k}\mathsf{in}\,j \mid \prod_{s\in\mathbf{T}_i^k}\omega_i\,[\mathsf{in}\,(s).\mathsf{out}\,(s^-).\mathsf{out}\,i.(\prod_{j\in\mathbf{S}_i^k}\mathsf{ls}_j\,[\mathsf{out}\,\omega_i])]].$$
$$\mathsf{N} \stackrel{def}{=} P_0 \mid \cdots \mid P_{k-1}.$$

In this solution, the process P_i consists of two parts: the ambient i that runs for the election and the ambient n_i that will stay put until the election is completed. After the declaration of the winner, the ambient ω_j will let out the ambients ls_j as acknowledgement of the loss of the other processes. These (the other processes) still exist because of the ambient $n_i\,[\mathsf{in}\,\mathsf{ls}_i]$. The latter will make a final acknowledgement of the loss, by entering the ambient ls_j. Thus, it is possible to formalise this. Consider a network N of size k such that once the winner is declared there is no interference of other processes. If there there exist $t, j \in \mathbb{N}$ such that $\mathsf{N} \longrightarrow_a \mathsf{N}^t \downarrow \omega_j$ then for some $t \leq t'$, for all $k \in \mathbf{S}_j^k$ we have $\mathsf{N}^{t'} \downarrow \mathsf{ls}_k$.

6.2.1.3 *Other ambient calculi*

One might ask if the positive results presented here hold for the other dialects of the ambient calculus presented here, PAC and SA. It is not difficult to see that the answer is positive. We write in this section the solution to the leader election problem for a network of size 2. First we present the solution for PAC.

PROPOSITION 6.2.21 In PAC^{pp} there exists a symmetric electoral system of size 2.

Proof. Let $\mathsf{N} \stackrel{def}{=} 0\,[\mathsf{pull}\,1.(\mathsf{push}\,\omega_0 \mid \omega_0\,[\,])] \mid 1\,[\mathsf{pull}\,0.(\mathsf{push}\,\omega_1 \mid \omega_1\,[\,])]$.
The first process to perform a pull wins.

For SAthe solution of electing a leader can be inferred from the solution of MA where the only capabilities used are in and out. Below, we shall present a different solution, since the only capabilities present are in and open unlike MA.

PROPOSITION 6.2.22 In SA^{iop} there exists a symmetric electoral system of size 2.

Proof. Consider the following network:

$$\mathsf{N} \stackrel{def}{=} (\mathsf{open}\,0 \mid 0\,[\mathsf{in}\,1 \mid \overline{\mathsf{in}}\,0.\overline{\mathsf{open}}\,0.\omega_0\,[\overline{\mathsf{open}}\,\omega_0]]) \mid (\mathsf{open}\,1 \mid 1\,[\mathsf{in}\,0 \mid \overline{\mathsf{in}}\,1.\overline{\mathsf{open}}\,1.\omega_1\,[\overline{\mathsf{open}}\,\omega_1])].$$

The first process to perform an in wins.

6.2.2 *Calculi without symmetric electoral systems*

In this subsection we are going to show that there are calculi that do not admit a symmetric electoral system. First of all, we shall reestablish Palamidessi's result on the π-calculus with separate choice, which states that π_s does not admit a symmetric electoral system (Theorem 4.2 of [71]). We shall prove that MA without in does not admit a symmetric electoral system.

6.2.2.1 *The π-calculus with separate choice*

In this section we prove that the π-calculus with separate choice does not solve the leader election problem in symmetric networks [71]. The idea of the proof is to show that there *exists* a computation that never breaks the initial symmetry, thus as noted also in Section §6.1 no process in the network can declare a unique winner. Our proof is different from Palamidessi's one in two ways:

- Palamidessi shows that there exists a computation where no winner is elected. We show that there exists a computation where either no winner is elected or every one is a winner.
- Palamidessi condensed her result in one statements only. We have partitioned the result in different statement for clarity.

First of all we prove some auxiliary lemmas.

LEMMA 6.2.23 Let P be a process in π_s and σ a permutation. Then $P \downarrow n$ if and only if $\sigma(P) \downarrow \sigma(n)$.

Proof. We prove the case for the output barb $P \downarrow \overline{n}$; the input barb case is similar. By definition of barbs, $P \downarrow \overline{n}$ if and only if $P \equiv_\pi (\nu p_1 \dots p_l)(\overline{n}\langle q\rangle.P' \mid P'')$ with $n \notin \{p_1 \dots p_l\}$ if and only if $\sigma(P) = (\nu p_1 \dots p_l)(\overline{\sigma(n)}\langle\sigma(q)\rangle.\sigma(P') \mid \sigma(P''))$ by definition of permutation, if and only if $\sigma(P) \downarrow \overline{\sigma(n)}$ by definition of barbs.

LEMMA 6.2.24 Let P be a process in π_s, σ a substitution and $Q{=}\sigma(P)$. Whenever $P \longrightarrow_\pi P'$ then for some Q', $Q \longrightarrow_\pi Q'$ and $Q'{=}\sigma(P')$.

Proof. By routine induction on $\longrightarrow_\pi$.

The following lemma says that initially a symmetric network cannot elect a leader. If one process declares a winner, everyone else is declared a winner as well.

LEMMA 6.2.25 Let $\mathsf{N} = [P_0 \mid \dots \mid P_{k-1}]$ be a symmetric network in π_s. If for some i such that $0 \le i \le k-1$ we have $\mathsf{N} \downarrow \overline{\omega}_i$, then for all l with $0 \le l \le k-1$ we have $\mathsf{N} \downarrow \overline{\omega}_l$.

Proof. Assume that N is symmetric with respect to σ with one orbit only. First of all it is important to observe that for such a σ for all $i \in \{0, 1, \ldots, k-1\}$

$$\mathcal{O}_\sigma(i) = \{i, \sigma^1(i) \ldots \sigma^{k-1}(i)\} = \{0, 1, \ldots, k-1\}.$$

If $\mathsf{N} \downarrow \overline{\omega}_i$ then there exists an r ($0 \leq r \leq k-1$) such that $P_r \downarrow \overline{\omega}_i$. By symmetry, for all h $P_{\sigma^h(r)}{=}\sigma^h(P_r)$ holds. Hence we have $P_{\sigma^h(r)} \downarrow \overline{\omega}_{\sigma^h(i)}$ by Lemma 6.2.23. By assumption, σ has one orbit, hence for all $j \in \{0, 1, \ldots, k-1\}$ it holds that

$$[P_r \mid P_{\sigma(r)} \mid \ldots \mid P_{\sigma^{k-1}(r)}] \downarrow \overline{\omega}_j.$$

The next lemma is crucial, it shows that for a symmetric network in π_s there exists a computation that never breaks the initial symmetry.

LEMMA 6.2.26 Let $\mathsf{N} = [P_0 \mid \ldots \mid P_{k-1}]$ be a symmetric network in π_s. Assume that $\mathsf{N} \longrightarrow_\pi \mathsf{N}^1$. Then there exists a computation $\mathcal{C} : \mathsf{N}^1 \longrightarrow\!\!\!\!\!\rightarrow_\pi \mathsf{N}^k$ such that:

i) N^k is symmetric;

ii) if for some i we have $\mathsf{N}^1 \downarrow \overline{\omega}_i$ and $\mathsf{N} \not\downarrow \overline{\omega}_i$, then for all h such that $0 \leq h \leq k-1$ we have $\mathsf{N}^k \downarrow \overline{\omega}_h$;

iii) if for all j $0 \leq j \leq k-1$ we have $\mathsf{N}^1 \not\downarrow \overline{\omega}_j$, then for all t, i such that $1 \leq t \leq k-1$ and $0 \leq i \leq k-1$ we have $\mathsf{N}^t \not\downarrow \overline{\omega}_i$.

Proof. Assume that N is a network of size k symmetric with respect to an automorphism σ with one orbit only and that $\mathsf{N} \longrightarrow_\pi \mathsf{N}^1$. There are two cases to consider:

Transition is derived from one process only : $\mathsf{N} \longrightarrow_\pi \mathsf{N}^1$ is the result of the computation of one process in the network. The process P_r belongs to N that has reduced (as follows) $P_r \longrightarrow_\pi P_r^*$.

Before proceeding with the proof, we remind the reader that by symmetry the following statements hold.

$$[P_0 \mid P_1 \mid \ldots \mid P_{k-1}] \equiv_\pi [P_r \mid P_{\sigma(r)} \ldots \mid P_{\sigma^{k-1}(r)}]$$

$$\begin{array}{rcll} P_{\sigma(r)} & = & \sigma(P_r) & \\ P_{\sigma^2(r)} & = & \sigma^2(P_r) & {=}\sigma(P_{\sigma(r)}) \\ P_{\sigma^3(r)} & = & \sigma^3(P_r) & {=}\sigma(P_{\sigma^2(r)}) \\ & \vdots & & \\ P_{\sigma^k(r)} & = & \sigma^k(P_r) & {=}\sigma(P_{\sigma^{k-1}(r)}) \end{array}$$

i) : If $P_r \longrightarrow_\pi P_r^*$ and by symmetry $\sigma(P_r) = P_{\sigma(r)}$ then by Lemma 6.2.24 for some $P^*_{\sigma(r)}$ $P_{\sigma(r)} \longrightarrow_\pi P^*_{\sigma(r)}$ and $P^*_{\sigma(r)} = \sigma(P_r^*)$. By repeating the previous reasoning for each process in the network, we conclude there are the following $k-1$ reductions.

$$\begin{array}{rcl} P_{\sigma(r)} & \longrightarrow_\pi & P^*_{\sigma(r)}{=}\sigma(P_r^*) \\ P_{\sigma^2(r)} & \longrightarrow_\pi & P^*_{\sigma^2(r)}{=}\sigma^2(P_r^*) \\ & \vdots & \\ P_{\sigma^{k-1}(r)} & \longrightarrow_\pi & P^*_{\sigma^{k-1}(r)}{=}\sigma^{k-1}(P_r^*) \end{array}$$

Therefore starting from the network $\mathsf{N}^1 = [P_r^* \mid P_{\sigma(r)} \mid \ldots \mid P_{\sigma^{k-1}(r)}]$ there exists the following computation.

$$\begin{array}{rll} \mathsf{N}^1: & [P_r^* \mid P_{\sigma(r)} \mid P_{\sigma^2(r)} \mid \ldots \mid P_{\sigma^{k-1}(r)}] & \longrightarrow_\pi \\ \mathsf{N}^2: & [P_r^* \mid P^*_{\sigma(r)} \mid P_{\sigma^2(r)} \mid \ldots \mid P_{\sigma^{k-1}(r)}] & \longrightarrow_\pi \\ \mathsf{N}^3: & [P_r^* \mid P^*_{\sigma(r)} \mid P^*_{\sigma^2(r)} \mid \ldots \mid P_{\sigma^{k-1}(r)}] & \longrightarrow_\pi \\ & & \vdots \\ \mathsf{N}^k: & [P_r^* \mid P^*_{\sigma(r)} \mid P^*_{\sigma^2(r)} \mid \ldots \mid P^*_{\sigma^{k-1}(r)}] & \end{array}$$

Hence $\mathsf{N}^1 \longrightarrow\!\!\!\!\!\rightarrow_\pi \mathsf{N}^k$. Moreover by symmetry we have:

$$[P_r^* \mid P^*_{\sigma(r)} \mid P^*_{\sigma^2(r)} \mid \ldots \mid P^*_{\sigma^{k-1}(r)}] \equiv_\pi [P_0^* \mid P_1^* \mid \ldots \mid P^*_{k-1}]$$

from which we conclude that N^k is symmetric with respect to σ. Since σ has not changed, it has one orbit only.

ii) : If for some h such that $0 \leq h \leq k-1$ we have $\mathsf{N} \not\downarrow \overline{\omega}_h$ and $\mathsf{N}^1 \downarrow \overline{\omega}_h$, it is the case that $\overline{\omega}_h$ has appeared during the first step of computation as follows.

$$P_r \longrightarrow_\pi P_r^* \downarrow \overline{\omega}_h$$

By Lemma 6.2.23 the remaining processes will exhibit the other barbs as follows:

$$\begin{array}{rcl} P_{\sigma(r)} & \longrightarrow_\pi & \sigma(P_r^*) \downarrow \overline{\omega}_{\sigma(h)} \\ P_{\sigma^2(r)} & \longrightarrow_\pi & \sigma^2(P_r^*) \downarrow \overline{\omega}_{\sigma^2(h)} \\ & \vdots & \\ P_{\sigma^{k-1}(r)} & \longrightarrow_\pi & \sigma^{k-1}(P_r^*) \downarrow \overline{\omega}_{\sigma^{k-1}(h)}. \end{array}$$

Hence, since σ has one orbit only, for all j such that $0 \leq j \leq k-1$ we have $\mathsf{N}^k \downarrow \overline{\omega}_j$.

iii) : If for all j such that $0 \leq j \leq k-1$ we have $\mathsf{N}^1 \not\downarrow \overline{\omega}_j$ then $P_r \longrightarrow_\pi P_r^* \not\downarrow \overline{\omega}_j$. Now assume for a contradiction that for some h and m $(0 \leq m, h \leq k-1)$ $\mathsf{N}^m \downarrow \omega_{\sigma^{m-1}(h)}$ and $\mathsf{N}^{m-1} \not\downarrow \omega_{\sigma^{m-1}(h)}$. Then there exists a process such that $P^*_{\sigma^{m-1}(r)} \downarrow \omega_{\sigma^{m-1}(h)}$ and by Lemma 6.2.23 $P_r \longrightarrow_\pi P_r^* \downarrow \overline{\omega}_h$, which is contradicts our previous assumption.

Transition comes from the communication between two processes : In this case the computation has occurred between two different processes in the network. We assume, without loss of generality and to the mere end of simplifying the notation, that P_0 and P_d are the two processes involved in the communication using the channel x. Moreover, we assume without loss of generality that P_d is the output and P_0 is the input, hence $P_0 \downarrow x$ and $P_d \downarrow \overline{x}$.

In order to simplify the notation, we assume that

$$\sigma(0) = 1 \;\; \sigma^2(0) = 2 \ldots \sigma^d(0) = d \ldots \sigma^{k-1}(0) = k-1 \;\; \sigma^k(0) = 0.$$

In the following proof we silently make use of structural congruence to move the processes to adjacent positions in order for them to reduce. In the same way they can be moved back after the reduction. Before carrying on with the proof, it is important to observe that symmetry yields a number of observables. In fact from $P_d \downarrow \overline{x}$, by symmetry and Lemma 6.2.23 we can deduce the observables for the other processes in the network as follows:

$$P_{\sigma(d)} \downarrow \sigma(\overline{x}) \;\; P_{\sigma^2(d)} \downarrow \sigma^2(\overline{x}) \ldots\ldots P_{\sigma^{k-1}(d)} \downarrow \sigma^{k-1}(\overline{x}).$$

Similarly since $P_0 \downarrow x$ then by symmetry and Lemma 6.2.23 we can deduce the observable for the other processes in the network as follows:

$$P_{\sigma(0)} \downarrow \sigma(x) \;\; P_{\sigma^2(0)} \downarrow \sigma^2(x) \ldots\ldots P_{\sigma^{k-1}(0)} \downarrow \sigma^{k-1}(x).$$

The syntactic form of the term P_0 is the following;

$$P_0 \equiv_\pi (\nu n_1^0 \ldots n_m^0)((x(z).P_0' + M_0^I) \mid (\overline{\sigma^{-d}(x)}\langle\sigma^{-d}(q)\rangle.P_0'' + M_0^O) \mid P_0''').$$

The syntactic form of the term P_d is the following:

$$P_d \equiv_\pi (\nu n_1^d \ldots n_m^d)((\sigma^d(x)(z).P_d' + M_d^I) \mid (\overline{x}\langle q\rangle.P_d'' + M_d^O) \mid P_d''').$$

There are two cases to consider:

a) if the output q is not bound, i.e. $q \notin \{n_1^d \ldots n_m^d\}$ then

$$P_0 \mid P_d \longrightarrow_\pi P_0^+ \mid P_d^-$$

where

$$P_0^+ = (\nu n_1^0 \ldots n_m^0)(P_0'\{q/z\} \mid (\overline{\sigma^{-d}(x)}\langle\sigma^{-d}(q)\rangle.P_0'' + M_0^O) \mid P_0''')$$

(the plus '+' as subscript means that the output is still present) and

$$P_d^- = (\nu n_1^d \ldots n_m^d)(\sigma^d(x)(z).P_d' + M_d^I) \mid P_d'' \mid P_d''')$$

(the plus '-' as subscript means that the output is not longer present).

i) : Now in general, for all s such that $0 \leq s \leq k-1$ we have

$$P_s \equiv_\pi (\nu n_1^s \ldots n_m^s)((\sigma^s(x)(z).P'_s + M_s^I) \mid (\overline{\sigma^{s-d}(x)}\langle\sigma^{s-d}(q)\rangle.P''_s + M_s^O) \mid P'''_s).$$

The residues of this process after computation are:

$$P_s^+ \equiv_\pi (\nu n_1^s \ldots n_m^s)(P'_s\{\sigma^s(q)/z\} \mid (\overline{\sigma^{s-d}(x)}\langle\sigma^{s-d}(q)\rangle.P''_s + M_s^O) \mid P'''_s)$$

and

$$P_s^- \equiv_\pi (\nu n_1^s \ldots n_m^s)((\sigma^s(x)(z).P'_s + M_s^I) \mid P''_s \mid P'''_s).$$

Moreover, if both input and output have been consumed we write as follows

$$P_s^{+-} = P_s^{-+} = (\nu n_1^s \ldots n_m^s)(P'_s\{\sigma^s(q)/z\} \mid P''_s \mid P'''_s).$$

Let $r = k - d$. Hence after r steps the P_0 contributes for the second time to the computation with an output. In order to show concretely how the computation goes around, we have to consider the relationship between r and d. There are two cases to consider: either $r < d$ or $r \geq d$. We analyse the case $r < d$ in detail; the other case is similar. If $r < d$ then for some m we have $d - r = m$. Hence the steps of reduction among the different processes in the network proceed as follows:

$$\begin{array}{rcl}
P_1 \mid P_{d+1} & \longrightarrow_\pi & P_1^+ \mid P_{d+1}^- \\
P_2 \mid P_{d+2} & \longrightarrow_\pi & P_2^+ \mid P_{d+2}^- \\
\vdots & & \\
P_{r-1} \mid P_{d+(r-1)} & \longrightarrow_\pi & P_{r-1}^+ \mid P_{d+(r-1)}^- \\
P_r \mid P_0^+ & \longrightarrow_\pi & P_r^+ \mid P_0^{+-} \\
P_{r+1} \mid P_1^+ & \longrightarrow_\pi & P_{r+1}^+ \mid P_1^{+-} \\
\vdots & & \\
P_{r+(m-1)} \mid P_{m-1}^+ & \longrightarrow_\pi & P_{r+(m-1)}^+ \mid P_{m-1}^{+-} \\
P_d^- \mid P_m^+ & \longrightarrow_\pi & P_d^{-+} \mid P_m^{+-} \\
\vdots & & \\
P_{d+(r-1)}^- \mid P_{m+(r-1)}^+ & \longrightarrow_\pi & P_{d+(r-1)}^{-+} \mid P_{m+(r-1)}^{+-}
\end{array}$$

Then, we can think that the network is partitioned in this way:

$$\mathsf{N} = [P_0 \mid \ldots \mid P_r \mid \ldots \mid P_d \mid \ldots \mid P_{d+r}]$$

The computation of the network is the following:

$$\begin{array}{lll}
\mathsf{N}^1: & [P_0^+ \mid \ldots \mid P_r \mid \ldots \mid P_d^- \mid \ldots \mid P_{k-1}] & \longrightarrow_\pi \\
\mathsf{N}^2: & [P_0^+ \mid P_1^+ \mid \ldots \mid P_r \mid \ldots \mid P_d^- \mid P_{d+1}^- \mid \ldots \mid P_{k-1}] & \longrightarrow_\pi \\
 & & \vdots \\
\mathsf{N}^r: & [P_0^{+-} \mid P_1^+ \mid \ldots \mid P_r^+ \mid \ldots \mid P_d^- \mid P_{d+1}^- \mid \ldots \mid P_{k-1}^-] & \longrightarrow_\pi \\
 & & \vdots \\
\mathsf{N}^k: & [P_0^{+-} \mid P_1^{+-} \mid \ldots \mid P_r^{+-} \mid \ldots \mid P_d^{+-} \mid P_{d+1}^{-+} \mid \ldots \mid P_{k-1}^{-+}]. &
\end{array}$$

Hence, we have $\mathsf{N}^1 \longrightarrow\!\!\!\!\rightarrow_\pi \mathsf{N}^k$.

Now we have to show that for the network N^k there is a permutation that is symmetric and has one orbit only. Clearly, σ is the permutation that we are looking for. It has one orbit only since it has not changed and is symmetric since for all s the following holds:

$$\begin{aligned}
P_s^{+-} &= (\nu n_1^s \ldots n_m^s)(P_s'\{\sigma^s(q)/z\} \mid P_s'' \mid P_s''') \\
&= (\nu n_1^s \ldots n_m^s)(\sigma^s(P_0'\{q/z\}) \mid \sigma^s(P_0'') \mid \sigma^s(P_0''')) \\
&= \sigma^s((\nu n_1^0 \ldots n_m^0)(P_0'\{q/z\} \mid P_0'' \mid P_0''')) \\
&= \sigma^s(P_0^{+-}).
\end{aligned}$$

ii) : If for some j $0 \leq j \leq k-1$ we have $\mathsf{N}^1 \downarrow \overline{\omega}_j$ and $\mathsf{N} \not\downarrow \overline{\omega}_j$, then either $P_0^+ \downarrow \overline{\omega}_j$ or $P_d^- \downarrow \overline{\omega}_j$. We consider $P_0^+ \downarrow \overline{\omega}_j$ (the other case is similar). Then $P_0^{+-} \downarrow \overline{\omega}_j$ from which we conclude by Lemma 6.2.23 for all s we have $P_{\sigma^s(0)}^{+-} \downarrow \overline{\omega}_{\sigma^s(j)}$. Hence for all h such that $0 \leq h \leq k$ we have $\mathsf{N}^k \downarrow \overline{\omega}_h$.

iii) : For all j $(0 \leq j \leq k-1)$ we have $\mathsf{N}^1 \not\downarrow \omega_j$ if and only if $P_0^+ \not\downarrow \overline{\omega}_j$ and $P_d^- \not\downarrow \overline{\omega}_j$. With a reasoning similar to the previous case, we prove that for all t, j such that $1 \leq t \leq k-1$ and $0 \leq j \leq k-1$ we have $\mathsf{N}^t \not\downarrow \overline{\omega}_j$.

b) If the output q is bound, i.e. $q \in \{n_1^d \ldots n_m^d\}$ then $q = n_z^d$ for some z such that $1 \leq z \leq m$.

i) :

$$P_0 \mid P_d \longrightarrow_\pi (\nu n_z^d)(P_0^+ \mid P_d^-)$$

where

$$\begin{aligned}
P_0^+ &= (\nu n_1^0 \ldots n_m^0)(P_0'\{n_z^d/w\} \mid (\overline{\sigma^{-d}(x)}\langle n_z^0\rangle.P_0'' + M_0^O) \mid P_0''') \\
P_d^- &= (\nu n_1^d \ldots n_{z-1}^d, n_{z+1}^d \ldots n_m^d)((\sigma^d(x)(w).P_d' + M_d^I) \mid P_d'' \mid P_d''')
\end{aligned}$$

In general then, for all s such that $0 \leq s \leq k-1$ the syntactical forms of the residues

of the computation are:

$$\begin{aligned} P_s^{+} &= (\nu n_1^s \ldots n_m^s)(P_s'\{n_z^s/w\} \mid (\overline{\sigma^{s-d}(x)}\langle n_z^s\rangle.P_s'' + M_s^O) \mid P_s''') \\ P_s^{-} &= (\nu n_1^s \ldots n_{z-1}^s n_{z+1}^s \ldots n_m^s)((\sigma^s(x)(w).P_s' + M_s^I \mid P_s'') \mid P_s''') \\ P_s^{-+} &= P_s^{+-} = (\nu n_1^s \ldots n_{z-1}^s n_{z+1}^s \ldots n_m^s)(P_s'\{n_z^s/w\} \mid P_s'' \mid P_s''') \end{aligned}$$

As in the previous case it is possible to show concretely how the computation goes around. Let $d = k - r$; then we have to consider the relationship between r and d. Again there are two cases, depending on whether $r < d$ or $d \leq r$. Assume that $r < d$ (the other case is similar) in which case $d - r = m$. The computation proceeds as shown below:

$$\begin{aligned} P_1 \mid P_{d+1} &\longrightarrow_\pi (\nu n_z^{d+1})(P_1^{+} \mid P_{d+1}^{-}) \\ P_2 \mid P_{d+2} &\longrightarrow_\pi (\nu n_z^{d+2})(P_2^{+} \mid P_{d+2}^{-}) \\ &\vdots \\ P_{r-1} \mid P_{d+(r-1)} &\longrightarrow_\pi (\nu n_z^{d+r-1})(P_{r-1}^{+} \mid P_{d+(r-1)}^{-}) \\ P_r \mid P_0^{+} &\longrightarrow_\pi (\nu n_z^{d+r})(P_r^{+} \mid P_0^{+-}) \\ P_{r+1} \mid P_1^{+} &\longrightarrow_\pi (\nu n_z^{1})(P_{r+1}^{+} \mid P_1^{+-}) \\ &\vdots \\ P_{r+(m-1)} \mid P_{m-1}^{+} &\longrightarrow_\pi (\nu n_z^{m-1})(P_{r+(m-1)}^{+} \mid P_{m-1}^{+-}) \\ P_d^{-} \mid P_m^{+} &\longrightarrow_\pi (n_z^{m})(P_d^{-+} \mid P_m^{+-}) \\ &\vdots \\ P_{d+(r-1)}^{-} \mid P_{m+(r-1)}^{+} &\longrightarrow_\pi (\nu n_z^{d-1})(P_{d+(r-1)}^{-+} \mid P_{m+(r-1)}^{+-}) \end{aligned}$$

The network can be seen partitioned in this way:

$$\mathsf{N} = [P_0 \mid \ldots \mid P_r \mid \ldots \mid P_d \mid \ldots \mid P_{d+(r-1)}].$$

The computation over the network extrudes the bound names which become global.

$$\begin{aligned} \mathsf{N}^1 : \quad & (\nu n_z^d)(P_0^{+} \mid \ldots \mid P_r \mid \ldots \mid P_d^{-} \mid \ldots \mid P_{k-1}) && \longrightarrow_\pi \\ & && \vdots \\ \mathsf{N}^k : \quad & (\nu n_z^0 \ldots, n_z^{k-1})(P_0^{+-} \mid P_1^{+-} \mid \ldots \mid P_r^{+-} \mid \ldots \mid P_d^{-+} \mid \ldots \mid P_{k-1}^{-+}) \end{aligned}$$

Hence, $\mathsf{N}^1 \longrightarrow\!\!\!\!\rightarrow_\pi \mathsf{N}^k$. Now, N^k is symmetric with respect to σ, since for all s ($0 \leq s \leq k-1$) the following holds:

$$\begin{aligned} P_s^{+-} &= (\nu n_1^s \ldots n_{z-1}^s n_{z+1}^s \ldots n_m^s)(P_s'\{n_z^s/w\} \mid P_s'' \mid P_s''') \\ &= \sigma^s((\nu n_1^0 \ldots n_{z-1}^0 n_{z+1}^s \ldots n_m^0)(P_0'\{n_z^0/w\} \mid P_0'' \mid P_0''')) \\ &= \sigma^s(P_0^{+-}). \end{aligned}$$

However, the scope extrusion of names n_z^s requires us to enlarge the domain of σ. We define σ' as follows

$$\sigma'(v) = \begin{cases} n_z^{q+1} & \text{if } v = n_z^q \text{ and } 0 \leq q \leq k-1 \\ \sigma(v) & \text{otherwise} \end{cases}$$

We require that network N^k is symmetric with respect to σ' and σ' has one orbit only. Observing that $\sigma' = \sigma \cup \{n_z^0 \mapsto n_z^1 \ldots n_z^{k-1} \mapsto n_z^0\}$ and that by symmetry for all h $(0 \leq l \leq k-1)$ we have $\sigma^h(P_0^{+-}) = P^{+-}_{\sigma^h(0)}$ then:

$$\begin{aligned} \sigma'^h(P_0^{+-}) &= \sigma^h(P_0^{+-})\{n_z^1/n_z^0\} \ldots \{n_z^h/n_z^{h-1}\} \\ &= P^{+-}_{\sigma^h(0)}\{n_z^1/n_z^0\} \ldots \{n_z^h/n_z^{h-1}\} \\ &= P^{+-}_{\sigma'^h(0)} \end{aligned}$$

which proves that σ' is symmetric. Since for all $i \in \mathsf{IN}$ we have $\sigma(i) = \sigma'(i)$, then it follows that σ' has one orbit only.

ii) : If for some j $0 \leq j \leq k-1$ we have $\mathsf{N}^1 \downarrow \overline{\omega}_j$ and $\mathsf{N} \not\downarrow \overline{\omega}_j$, then either $P_0^+ \downarrow \overline{\omega}_j$ or $P_d^- \downarrow \overline{\omega}_j$. By reasoning similarly to the previous case on σ', it is the case that for all l such that $0 \leq l \leq k-1$ we have $\mathsf{N}^k \downarrow \overline{\omega}_l$.

iii) : For all j such that $0 \leq j \leq k$ we have $\mathsf{N}^1 \not\downarrow \omega_j$ if and only if $P_0^+ \not\downarrow \overline{\omega}_j$ and $P_d^- \not\downarrow \overline{\omega}_j$. With a reasoning similar to the previous case, we prove that for all t, j such that $1 \leq t \leq k-1$ and $0 \leq j \leq k-1$ we have $\mathsf{N}^t \not\downarrow \overline{\omega}_j$.

So far we have shown that starting from a symmetric state, a network of size k reaches in k-steps of computation another symmetric state, where either everyone is a winner or nobody has won the election. In both cases the election for the leader has failed. It remains to show that for any maximal computation, the property described above is preserved. The next lemma expresses exactly this.

LEMMA 6.2.27 Let $\mathsf{N} = [P_0 \mid \ldots \mid P_{k-1}]$ with $k \geq 2$ be a symmetric network in π_s. Then there exists a maximal computation $\mathcal{C}$ such that $\mathsf{Obs}(\mathcal{C}) = \{\omega_0, \omega_1, \ldots, \omega_j, \ldots, \omega_{k-1}\}$ or $\mathsf{Obs}(\mathcal{C}) = \emptyset$.

Proof. The proof goes by induction on $n \in \mathsf{IN}$ where n is the number of steps of the computation $\mathsf{N} \longrightarrow\!\!\!\!\!\rightarrow_\pi \mathsf{N}^n$.

$n = 0$: The network N initially either does not display any winner i.e. for all j such that $0 \leq j \leq k-1$ we have $\mathsf{N} \not\downarrow \omega_j$, or if for some i we have $\mathsf{N} \downarrow \overline{\omega}_i$, then by Lemma 6.2.25 for all h such that $0 \leq h \leq k-1$ we have $\mathsf{N} \downarrow \overline{\omega}_h$.

$n = t$: Assume by induction hypothesis there exists a $\mathcal{C}$ with $t' \leq t$ such that $\mathcal{C} : \mathsf{N} \longrightarrow\!\!\!\!\!\rightarrow_{\pi} \mathsf{N}^{t'}$ with $\mathsf{N}^{t'}$ symmetric with respect to an automorphism σ. Such a computation is either finite, i. e. it has stopped $\mathsf{N}^{t'} \not\longrightarrow_{\pi}$ then and $t' = t$, or the computation proceeds $\mathsf{N}^{t'} \longrightarrow\!\!\!\!\!\rightarrow_{\pi}$. Moreover, either there is more than one winner i.e. $\mathsf{Obs}(\mathcal{C}) = \{\omega_0, \omega_1, \ldots \omega_{k-1}\}$, or there is no winner i.e. $\mathsf{Obs}(\mathcal{C}) = \emptyset$.

There are two cases to consider:

a) If $\mathsf{N}^{t'} \not\rightarrow$ and either $\mathsf{Obs}(\mathcal{C}) = \{\omega_0, \omega_1, \ldots \omega_{k-1}\}$ or $\mathsf{Obs}(\mathcal{C}) = \emptyset$ then the Lemma is proved.

b) Otherwise, it is the case that $\mathcal{C} : \mathsf{N}^{t'} \longrightarrow\!\!\!\!\!\rightarrow_{\pi}$ and $\mathsf{Obs}(\mathcal{C}) = \{\omega_0, \omega_1, \ldots \omega_{k-1}\}$ then the Lemma is proved. Otherwise $\mathcal{C} : \mathsf{N}^{t'} \longrightarrow_{\pi}$ and $\mathsf{Obs}(\mathcal{C}) = \emptyset$, then there are two cases to consider:

1) Assume $\mathsf{N}^{t'} \longrightarrow_{\pi} \mathsf{N}^{t'+1}$ and for some i we have $\mathsf{N}^{t'+1} \downarrow \overline{\omega}_i$. Then by Lemma 6.2.26 there exists a computation $\mathcal{C}' : \mathsf{N}^{t'+1} \longrightarrow_{\pi} \mathsf{N}^{t'+k}$ such that, for all j such that $0 \leq j \leq k$ we have $\mathsf{N}^{t'+k} \downarrow \overline{\omega}_j$. Hence there exists a computation $\mathcal{C}'' : \mathsf{N} \longrightarrow\!\!\!\!\!\rightarrow_{\pi} \mathsf{N}^{t'+k}$ such that $\mathsf{Obs}(\mathcal{C}'') = \{\omega_0, \omega_1, \ldots, \omega_j, \ldots, \omega_{k-1}\}$.

2) Assume $\mathsf{N}^{t'} \longrightarrow\!\!\!\!\!\rightarrow_{\pi} \mathsf{N}^{t'+1}$ and for all i we have $\mathsf{N}^{t'+1} \not\downarrow \overline{\omega}_i$. Then by Lemma 6.2.26 there exists a computation $\mathcal{C}' : \mathsf{N}^{t'+1} \longrightarrow\!\!\!\!\!\rightarrow_{\pi} \mathsf{N}^{t'+k}$ such that $\mathsf{N}^{t'+k} \not\downarrow \overline{\omega}_i$. Hence there exists a computation $\mathcal{C}'' : \mathsf{N} \longrightarrow\!\!\!\!\!\rightarrow_{a} \mathsf{N}^{t'+k}$ such that $\mathsf{Obs}(\mathcal{C}'') = \emptyset$.

The following theorem claims that there does not exist a symmetric electoral system of any finite size in π_s. The statement is identical to Palamidessi's. For the proof, previous results in this section are used.

THEOREM 6.2.28 [71] Let $\mathsf{N} = [P_0 \mid \ldots \mid P_{k-1}]$ with $k \geq 2$ be a symmetric network in π_s. Then N cannot be an electoral system.

Proof. Assume for a contradiction that N is an electoral system. Then for every maximal computation $\mathcal{C}$ a winner has to be elected. Therefore $\mathsf{Obs}(\mathcal{C})$= $\{\omega_i\}$ with $0 \leq i \leq k-1$. By Lemma 6.2.27 there exists a maximal computation $\mathcal{C}$ such that either $\mathsf{Obs}(\mathcal{C}) = \{\omega_0, \omega_1, \ldots, \omega_j, \ldots, \omega_{k-1}\}$ or $\mathsf{Obs}(\mathcal{C}) = \emptyset$. In either case N cannot be an electoral system.

6.2.2.2 *The Ambient Calculus without the* in *capability*

We have shown in Subsection 6.2.1 that MA with the in and out capabilities only, can solve the leader election problem in a symmetric network. This kind of problem can be solved in the

π-calculus with mixed choice, but not in the version with separate choice. It is clear that the mixed choice operator is the key for the expressiveness result.

One question is which operator makes a difference in expressiveness for ambients. This question is also interesting for the following reason. It might be argued that leader election problems are not interesting in the ambient setting, because of the inherent tree structure of processes. The next theorem will show that this is not completely true. The tree structure of ambients is not indeed the key to expressiveness. In fact, we could keep the tree structure and remove one capability, $\mathsf{in}\, n$, in which case the leader election problem cannot be solved. This means, in simple words, that for breaking symmetry the in capability is crucial.

Before proving the main theorem we need to show some auxiliary lemmas, similarly to the ones presented in the previous section.

DEFINITION 6.2.29

We define MA^{-in} to be MA without the in capability.

We shall show a few results similar to the ones shown before for the π-calculus with separate choice.

LEMMA 6.2.30 Let P be a process in MA^{-in} and σ a permutation. Then $P \downarrow n$ if and only if $\sigma(P) \downarrow \sigma(n)$.

Proof. By definition of barbs, $P \downarrow n$ if and only if $P \equiv_a (\nu p_1 \ldots p_l)(n\,[P'] \mid P'')$ with $n \notin \{p_1 \ldots p_l\}$ if and only if $\sigma(P) \equiv_a (\nu p_1 \ldots p_l)(\sigma(n)\,[\sigma(P')] \mid \sigma(P''))$ by definition of permutation, if and only if $\sigma(P) \downarrow \sigma(n)$ by definition of barbs.

LEMMA 6.2.31 Let P be a process in MA^{-in}, σ a substitution. and $Q{=}\sigma(P)$. If $P \longrightarrow_a P'$, then for some Q', $Q \longrightarrow_a Q'$ and $Q'{=}\sigma(P')$.

Proof. By induction on $\longrightarrow_a$.

The following lemma, similarly to Lemma 6.2.25. says that initially a symmetric network cannot elect a leader. If one process declares a winner, everyone else is declared a winner as well.

LEMMA 6.2.32 Let $\mathsf{N} = [P_0 \mid \ldots \mid P_{k-1}]$ be a symmetric network in MA^{-in}. If for some i such that $0 \leq i \leq k-1$ we have $\mathsf{N} \downarrow \omega_i$, then for all l with $0 \leq l \leq k-1$ we have $\mathsf{N} \downarrow \omega_l$.

Proof. The proof is identical to that of Lemma 6.2.25.

LEMMA 6.2.33 Let $\mathsf{N} = [P_0 \mid \ldots \mid P_{k-1}]$ be a symmetric network in MA^{-in}. Assume that $\mathsf{N} \longrightarrow_a \mathsf{N}^1$. Then there exists a computation $\mathcal{C} : \mathsf{N}^1 \longrightarrow\!\!\!\!\rightarrow_a \mathsf{N}^k$ such that:

i) N^k is symmetric;

ii) if for some i we have $\mathsf{N}^1 \downarrow \omega_i$ and $\mathsf{N} \not\downarrow \omega_i$, then for all h such that $0 \leq h \leq k-1$ we have $\mathsf{N}^k \downarrow \omega_h$;

iii) if for all j such that $0 \leq j \leq k-1$ we have $\mathsf{N}^1 \not\downarrow \omega_j$, then for all t, i such that $1 \leq t \leq k-1$ and $0 \leq i \leq k-1$ we have $\mathsf{N}^t \not\downarrow \omega_i$.

Proof. Assume that N of size k is symmetric with respect to an automorphism σ with one orbit only and that $\mathsf{N} \longrightarrow_a \mathsf{N}^1$. There are two cases to consider:

Computation derived from one process only : $\mathsf{N} \longrightarrow_a \mathsf{N}^1$ is the result of the computation of one process in the network. Assume P_r in N has reduced as follows $P_r \longrightarrow_a P_r^*$. The proof for this case is identical to that of Lemma 6.2.26.

Computation derived from the communication of two processes in the network : In this case the computation has occurred between two different processes in the network.

We assume, without loss of generality and to the mere end of simplifying the notation, that P_0 and P_d are the two processes involved in the computation. Moreover, in order to simplify the notation, we assume that

$$\sigma(0) = 1 \;\; \sigma^2(0) = 2 \ldots \sigma^d(0) = d \ldots \sigma^{k-1}(0) = k-1 \;\; \sigma^k(0) = 0$$

which implies by symmetry the following:

$$[\sigma(P_0) \mid \sigma^2(P_0) \mid \ldots \mid \sigma^{k-1}(P_0)] = [P_0 \mid P_{\sigma(0)} \ldots \mid P_{\sigma^{k-1}(0)}]$$

By the operational semantics of this limited calculus, there are two ways in which this calculus can reduce.

- Two processes in the network form a open-redex, i.e. they reduce by the use of the rule RED OPEN. We assume, without loss of generality that

$$\begin{aligned} P_0 &= (\nu p_1^0 \ldots p_s^0)(\mathsf{open}\, n.S_0 \mid P_0') \\ P_d &= (\nu p_1^d \ldots p_s^d)(n\,[Q_d] \mid P_d') \end{aligned}$$

with $n \notin \{p_1^0 \ldots p_s^0\} \cup \{v_1^d \ldots v_s^d\}$ (otherwise the two processes would not be able to reduce).

Moreover we will silently use structural congruence to move processes to adjacent positions in order to reduce and to move them back again to their original positions.

i) : By symmetry, we conclude the following statements starting from P_0.

$$\begin{aligned} P_{\sigma(0)} &= (\nu p_1^1 \ldots p_s^1)(\mathsf{open}\,\sigma(n).\sigma(S_0) \mid \sigma(P_0')) \\ P_{\sigma^2(0)} &= (\nu p_1^2 \ldots p_s^2)(\mathsf{open}\,\sigma^2(n).\sigma^2(S_0) \mid \sigma^2(P_0')) \\ &\vdots \\ P_{\sigma^{k-1}(0)} &= (\nu p_1^{k-1} \ldots p_s^{k-1})(\mathsf{open}\,\sigma^{k-1}(n).\sigma^{k-1}(S_0) \mid \sigma^{k-1}(P_0')) \end{aligned}$$

Similar conclusions can be drawn starting from P_d.

$$\begin{aligned} P_{\sigma(d)} &= (\nu p_1^{d+1} \ldots p_s^{d+1})(\sigma(n)\,[\sigma(Q_d)] \mid \sigma(P_d')) \\ P_{\sigma^2(d)} &= (\nu p_1^{d+2} \ldots p_s^{d+2})(\sigma^2(n)\,[\sigma^2(Q_d)] \mid \sigma^2(P_d')) \\ &\vdots \\ P_{\sigma^{k-1}(d)} &= (\nu p_d^{d-1} \ldots p_s^{d-1})(\sigma^{k-1}(n)\,[\sigma^{k-1}(Q_d)] \mid \sigma^{k-1}(P_d')) \end{aligned}$$

Hence, we conclude that each process in the network contains an open-redex. First, we analyse P_0 and P_d.

$$\begin{aligned} P_0 &= (\nu p_1^0 \ldots p_s^0)(\mathsf{open}\,n.S_0 \mid \sigma^{-d}(n)\,[Q_0] \mid R_0) \\ P_d &= (\nu p_1^d \ldots p_s^d)(\mathsf{open}\,\sigma^d(n).S_d \mid n\,[Q_d] \mid R_d) \end{aligned}$$

Now if $P_0 \mid P_d \longrightarrow_a P_0^+ \mid P_d^-$ then the residual processes must have the syntactical form:

$$\begin{aligned} P_0^+ &= (\nu p_1^0 \ldots p_s^0)(S_0 \mid \sigma^{-d}(n)\,[Q_0] \mid R_0) \\ P_d^- &= (\nu p_1^d \ldots p_s^d)(\mathsf{open}\,\sigma^d(n).S_d \mid Q_d \mid R_d). \end{aligned}$$

Therefore in general, for all h such that $0 \leq h \leq k-1$ we have

$$P_h = (\nu p_1^h \ldots p_s^h)(\mathsf{open}\,\sigma^h(n).S_h \mid \sigma^{h-d}(n)\,[Q_h] \mid R_h)$$

and

$$\begin{aligned} P_h^+ &= (\nu p_1^h \ldots p_s^h)(S_h \mid \sigma^{h-d}(n)\,[Q_h]R_h) \\ P_h^+ &= (\nu p_1^h \ldots p_s^h)(\mathsf{open}\,\sigma^h(n).S_h \mid Q_h \mid R_h) \\ P_h^{+-} &= P_d^{-+} = (\nu p_1^h \ldots p_s^h)(S_h \mid Q_h \mid R_h). \end{aligned}$$

Now in order to show concretely how the computation goes around (similarly to what has been done in the previous proof of Lemma 6.2.26), let $k - d = r$. Now we need to consider the relationship between r and d. There are two cases to consider: $r \leq d$ or $d < r$. We consider $d < r$ (the other case was considered in the previous proof of Lemma 6.2.26) in which case $r - d = m$ and assume that $k = 2d + m$ (the other

cases are similar). Hence the computation proceeds as shown below:

$$\begin{array}{rcl}
P_1 \mid P_{d+1} & \longrightarrow_a & P_1^+ \mid P_{d+1}^- \\
P_2 \mid P_{d+2} & \longrightarrow_a & P_2^+ \mid P_{d+2}^- \\
 & \vdots & \\
P_{d-1} \mid P_{2d-1} & \longrightarrow_a & P_{d-1}^+ \mid P_{2d-1}^- \\
P_d^- \mid P_{2d} & \longrightarrow_a & P_d^{-+} \mid P_{2d}^- \\
P_{d+1}^- \mid P_{2d+1}^+ & \longrightarrow_a & P_{d+1}^{-+} \mid P_{2d+1}^- \\
 & \vdots & \\
P_{d+(m-1)}^- \mid P_{k-1} & \longrightarrow_a & P_{d+(m-1)}^{-+} \mid P_{k-1}^- \\
P_{2d}^{-+} \mid P_0^+ & \longrightarrow_a & P_{d+m}^{-+} \mid P_0^{+-} \\
 & \vdots & \\
P_{k-1}^- \mid P_{d-1}^+ & \longrightarrow_a & P_{k-1}^{-+} \mid P_{d-1}^{+-}
\end{array}$$

Then the network is partitioned in this way:

$$\mathsf{N}^h = [P_0 \mid \ldots \mid P_d \mid \ldots \mid P_{2d} \mid \ldots \mid P_{k-1}].$$

The computation of the network after the initial step is the following:

$$\begin{array}{lcl}
\mathsf{N}^1: & [P_0^+ \mid P_1 \mid \ldots \mid P_d^- \mid P_{d+1} \ldots \mid P_{k-1}] & \longrightarrow_a \\
\mathsf{N}^2: & [P_0^+ \mid P_1^+ \mid \ldots \mid P_d^- \mid P_{d+1}^- \mid \ldots \mid P_{k-1}] & \longrightarrow_a \\
 & & \vdots \\
\mathsf{N}^d: & [P_0^+ \mid P_1^+ \mid \ldots \mid P_d^{-+} \mid \ldots \mid P_{2d}^- \mid \ldots \mid P_{k-1}] & \longrightarrow_a \\
 & & \vdots \\
\mathsf{N}^k: & [P_0^{+-} \mid P_1^{+-} \mid \ldots \mid P_d^{-+} \mid \ldots \mid P_{2d}^{-+} \mid \ldots \mid P_{k-1}^{-+}] &
\end{array}$$

Hence, $\mathsf{N}^1 \longrightarrow\!\!\!\!\!\rightarrow_a \mathsf{N}^k$.

Now we have to show that the network N^k is symmetric.

Now, N^k is symmetric with respect to σ, since for all s ($0 \leq s \leq k-1$) the following holds:

$$\begin{array}{rcl}
P_{\sigma^h(0)}^{+-} & = & (\nu p_1^h \ldots p_s^h)(S_h \mid Q_h \mid R_0)) \\
 & = & (\nu p_1^h \ldots p_s^h)(\sigma^h(S_0) \mid \sigma^h(Q_0) \mid \sigma^h(R_0)) \\
 & = & \sigma^h((\nu p_1^0 \ldots p_s^0)(S_0 \mid Q_0 \mid R_0)) \\
 & = & \sigma^h(P_0^{+-})
\end{array}$$

.

The automorphism σ has one orbit only since it has not changed.

ii) : If for some h such that $0 \leq h \leq k-1$ we have $\mathsf{N} \not\downarrow \omega_h$ and $\mathsf{N}^1 \downarrow \omega_h$, then either $P_0^+ \downarrow \omega_h$ or $P_d^- \downarrow \omega_h$. Let's consider $P_0^+ \downarrow \omega_h$ (the other case is similar), then we have $P_0^{+-} \downarrow \omega_h$ from which we conclude by Lemma 6.2.23 for all $s$we have $P^{+-}_{\sigma^s(0)} \downarrow \omega_{\sigma^s(h)}$. Hence, since σ has one orbit only, for all h such that $0 \leq h \leq k$ we have $\mathsf{N}^k \downarrow \omega_h$.

iii) : If for all j such that $0 \leq j \leq k-1$ we have $\mathsf{N}^1 \not\downarrow \omega_j$, then $P_0 \mid P_d \longrightarrow_a P_0^+ \mid P_d^- \not\downarrow \omega_j$ if and only if we have $P_0^+ \not\downarrow \omega_j$ and $P_d^- \not\downarrow \omega_j$. Now assume for a contradiction that for some h and t such that $0 \leq t, h \leq k-1$ we have $\mathsf{N}^t \downarrow \omega_{\sigma^{t-1}(h)}$ and $\mathsf{N}^{t-1} \not\downarrow \omega_{\sigma^{t-1}(h)}$. Then there are different cases to consider: $t < d$ or $d \leq t \leq 2d$ or $2d+1 \leq t \leq k-1$. If $t < d$ then we have $P_{\sigma^{t-1}(0)} \mid P_{\sigma^{t-1}(d)} \longrightarrow_a P^+_{\sigma^{t-1}(0)} \mid P^-_{\sigma^{t-1}(d)} \downarrow \omega_{\sigma^{t-1}(h)}$, which means by definition of barbs that either $P^+_{\sigma^{t-1}(0)} \downarrow \omega_{\sigma^{t-1}(h)}$ or $P^-_{\sigma^{t-1}(d)} \downarrow \omega_{\sigma^{t-1}(h)}$. We consider $P^+_{\sigma^{t-1}(0)} \downarrow \omega_{\sigma^{t-1}(h)}$ (the other case is similar), from which by Lemma 6.2.30 it holds that $P_0^+ \downarrow \omega_h$ which implies that $\mathsf{N}^1 \downarrow \omega_h$. This is a clear contradiction with the assumption. If $d \leq t \leq 2d$ then we have $P^-_{\sigma^{t-1}(0)} \mid P_{\sigma^{t-1}(d)} \longrightarrow_a P^{-+}_{\sigma^{t-1}(0)} \mid P^-_{\sigma^{t-1}(d)} \downarrow \omega_{\sigma^{t-1}(h)}$, which implies that either $P^{-+}_{\sigma^{t-1}(0)} \downarrow \omega_{\sigma^{t-1}(h)}$ or $P^-_{\sigma^{t-1}(d)} \downarrow \omega_{\sigma^{t-1}(h)}$. Now we consider $P^{-+}_{\sigma^{t-1}(0)} \downarrow \omega_{\sigma^{t-1}(h)}$ (the other case is similar to the previous one). Since we have $P^{-+}_{\sigma^{t-1}(0)} = (\nu p_1^{t-1} \ldots p_s^{t-1})(P_{\sigma^{t-1}(0)} \mid Q_{\sigma^{t-1}(0)} \mid R_{\sigma^{t-1}(0)})$ we conclude that either $P_{\sigma^{t-1}(0)} \downarrow \omega_{\sigma^{t-1}(h)}$ or $Q_{\sigma^{t-1}(0)} \downarrow \omega_{\sigma^{t-1}(h)}$. Assume that $P_{\sigma^{t-1}(0)} \downarrow \omega_{\sigma^{t-1}(h)}$ then by Lemma 6.2.30 then $P_0^+ \downarrow \omega_h$ which implies $\mathsf{N}^1 \downarrow \omega_h$. This contradict our previous assumption. On the other hand, if we have $Q_{\sigma^{t-1}(0)} \downarrow \omega_{\sigma^{t-1}(h)}$ then for some s we have $\sigma^s(d) = \sigma^{t-1}(0)$, hence $Q_{\sigma^s(d)} \downarrow \omega_{\sigma^{t-1}(h)}$. Therefore by Lemma 6.2.30 $Q_d \downarrow \omega_{\sigma^{(t-1)-s}(h)}$ and by definition of barbs $P_d^- \downarrow \omega_{\sigma^{(t-1)-s}(h)}$ and $\mathsf{N}^1 \downarrow \omega_{\sigma^{(t-1)-s}(h)}$, which is contradicts my our assumption.

The case $2d+1 \leq t \leq k-1$ is similar to the previous one.

- The reduction is triggered by the rule RED COMM, this means the redex is formed with communication primitives. Then we have:

$$\begin{aligned} P_0 &= (\nu p_1^0 \ldots p_s^0)(\langle n \rangle \mid S_0) \\ P_d &= (\nu p_1^d \ldots p_s^d)((y).Q_d \mid P'_d). \end{aligned}$$

Thus, with reasoning similar to the previous case, we can conclude that for each j such that $0 \leq j \leq k-1$:

$$P_j = (\nu p_1^j \ldots p_s^j)(\langle \sigma^j(n) \rangle \mid (y).Q_j \mid P'_j \mid S_j).$$

Thus, each process reduces independently as in the case of 'Computation derived from one process only'. The proof then is identical to that case.

LEMMA 6.2.34 Let $\mathsf{N} = [P_0 \mid \ldots \mid P_{k-1}]$ with $k \geq 2$ be a symmetric network in MA^{-in}. Then there exists a computation $\mathcal{C}$ such that either $\mathsf{Obs}(\mathcal{C}) = \{\omega_0, \omega_1, \ldots, \omega_j, \ldots, \omega_{k-1}\}$ or $\mathsf{Obs}(\mathcal{C}) = \emptyset$.

Proof. The proof is identical to that of Lemma 6.2.27.

THEOREM 6.2.35 Let $\mathsf{N} = [P_0 \mid \ldots \mid P_{k-1}]$ with $k \geq 2$ be a symmetric network in MA^{-in}. Then N cannot be an electoral system.

Proof. The proof is identical to that of Theorem 6.2.28 using Lemma 6.2.34.

6.2.2.3 *Other ambient calculi*

Before introducing the separation result we would like to point that negative results hold also for PAC and SA. Let's define SA^{-in} to be Safe Ambients without in capability, and PAC^{-pull} to be the push and pull ambient calculus without the pull capability. It should be quite clear that PAC^{-pull} and SA^{-in} do not admit an electoral system either. This can be proved quite easily by minor modification of the proof of Lemma 6.2.33.

6.3 *When do encodings not exist?*

In Section 3.1 we discussed when a function between two languages can be accepted as an encoding. We observed that we are not keen to accept any function between languages as an encoding, and that we expect some semantic properties to be preserved.

Perhaps surprisingly, one could to argue also to show that encodings do not exist, since a function between two languages can always be defined. We argue that in dealing with leader election problems, an encoding must: *preserve the fundamental criteria of the problem* and *not introduce a solution.* We present below the *conditions* for an encoding to preserve symmetric electoral systems, and we show the general result in Lemma 6.3.2. Afterwards we shall discuss this notion of encoding.

DEFINITION 6.3.1 Let L, L' be process languages. An encoding $[\![-]\!] : L \to L'$ is

i) *distribution-preserving* if for all processes P, Q of L, $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$.

ii) *permutation-preserving* if for any permutation of names σ in L there exists a permutation θ in L' such that $[\![\sigma(P)]\!] = \theta([\![P]\!])$ and the permutations are compatible, in that for all $i \in \mathbb{N}$ both $\sigma(i) = \theta(i)$ and $\sigma(\omega_i) = \theta(\omega_i)$.

iii) *observation-respecting* if for any P in L,

a) for every maximal computation $\mathcal{C}$ of P there exists a maximal computation $\mathcal{C}'$ of $[\![P]\!]$ such that $\mathsf{Obs}(\mathcal{C}) = \mathsf{Obs}(\mathcal{C}')$

b) for every maximal computation $\mathcal{C}$ of $[\![P]\!]$ there exists a maximal computation $\mathcal{C}'$ of P such that $\mathsf{Obs}(\mathcal{C}) = \mathsf{Obs}(\mathcal{C}')$

An encoding which preserves distribution and permutation is *uniform.*

The first two items in Definition 6.3.1 (i.e. uniformity) are as in Palamidessi [71]. The condition of preserving distribution is important in ruling out encodings which make use of a *central server.* The second condition prevents a trivial solution from being introduced by collapsing all the set of natural numbers $\{0, 1, \ldots, k-1\}$ to a $j \in \mathbb{N}$. Notice that the first two items aim to map symmetric networks to symmetric networks of the same size and with the same orbit. The third item aims to preserve the uniqueness of the winner. Because the winner in this framework is represented with a barb, the condition is on the semantics.

Bougé [12] defined an encoding as "reasonable" if it maps electoral systems to electoral systems. The condition of respecting observations is our interpretation of Palamidessi's requirement of "preserving a reasonable semantics" [71]. She states that a reasonable semantics should distinguish processes which differ on the observables of their maximal computations. In fact, we only require part (b) to ensure that electoral systems are mapped to electoral systems; part (a) is added to make the condition more natural. In their version of Palamidessi's work, Sangiorgi and Walker [78] use a condition that if the observables of every maximal computation of a process P are singletons, then the same is true for the encoding of P. This obviously relates very directly to the need to preserve electoral systems. Finally, Ene and Muntian [28] use yet another formulation. As it only refers to finite computations, it would not be enough for our purposes.

Symmetric electoral systems are mapped to symmetric electoral systems by encodings satisfying Definition 6.3.1:

LEMMA 6.3.2 Suppose $[\![-]\!] : L \to L'$ is a uniform observation-respecting encoding. Suppose that N is a symmetric electoral system of size k with no globally bound variables. Let $\mathsf{N}' \stackrel{def}{=} [\![\mathsf{N}]\!]$. Then N' is also a symmetric electoral system of size k.

Proof. Assume that the network $\mathsf{N} = P_0 \mid P_1 \mid \ldots \mid P_{k-1}$ of size k is an electoral system in L

and that $[\![-]\!] : L \to L'$ is a uniform observation-respecting encoding. We are going to show that $[\![P_0 \mid P_1 \mid \ldots \mid P_{k-1}]\!]$ is a symmetric electoral system, i.e. every maximal computation yields one winner only. Since $[\![-]\!]$ is distribution-preserving (Definition 6.3.1 (i))) then it preserves the size of the network:

$$[\![P_0 \mid P_1 \mid \ldots \mid P_{k-1}]\!] = [\![P_0]\!] \mid [\![P_1]\!] \mid\mid \ldots \mid [\![P_{k-1}]\!].$$

By symmetry for all i such that $0 \leq i \leq k-1$ we have $\sigma(P_i) = P_{\sigma(i)}$ and since $[\![-]\!]$ is permutation-preserving (Definition 6.3.1 (ii)), then there exists a θ such that for all $i \in \mathbb{N}$ we have $\theta(i) = \sigma(i)$.

$$\begin{aligned} \theta([\![P_i]\!]) &= [\![\sigma(P_i)]\!] && \text{by Definition } 6.3.1(ii) \\ &= [\![P_{\sigma(i)}]\!] && \text{by symmetry} \\ &= [\![P_{\theta(i)}]\!] && \text{since } \theta(i) = \sigma(i). \end{aligned}$$

Hence N' is symmetric with respect to θ (with one orbit only). It remains to show that N' is an electoral system. Consider a maximal computation $\mathcal{C}'$ of N'. By condition $iii(b)$ on Definition 6.3.1 there must exist a computation $\mathcal{C}$ of N such that $\mathsf{Obs}(\mathcal{C}) = \mathsf{Obs}(\mathcal{C}')$. Now since N is an electoral system, every maximal computation exhibits one winner only. Hence $\mathsf{Obs}(\mathcal{C}) = \{\omega_j\}$ for some j such that $0 \leq j \leq k-1$, which implies that $\mathsf{Obs}(\mathcal{C}') = \{\omega_j\}$. Since this is true for every maximal computation $\mathcal{C}'$ of N', the lemma is proven.

6.4 *Separation results*

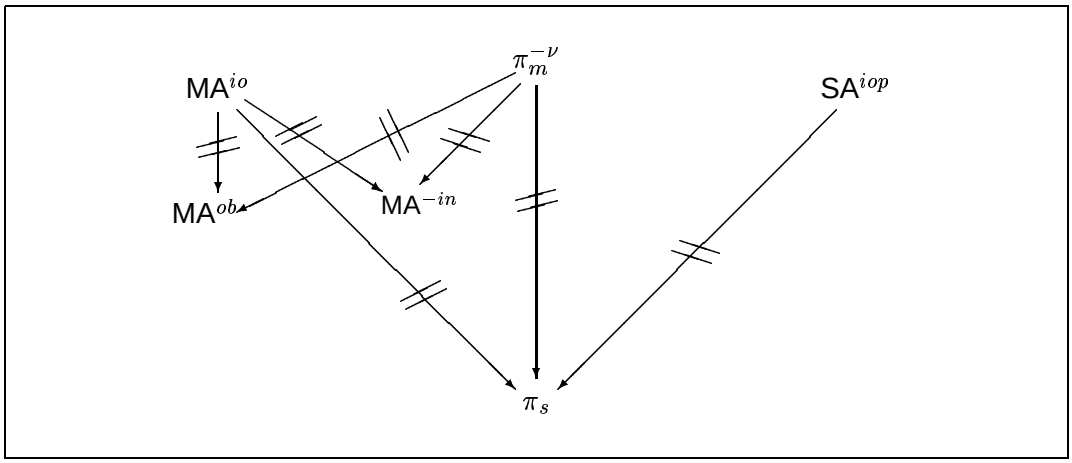
In this section, we present the separation results that can be derived from the work done so far in this dissertation. In Figure 6.1, we provide a graphical view of the separation results between the calculi and their dialects dealt with in this chapter.

6.4.1 *Negative results for the π-calculus*

First of all, we show that the π-calculus with mixed choice and without restriction cannot be encoded into the π-calculus with separate choice. This is a stronger version of Palamidessi's result [71] (Corollary 7.1).

COROLLARY 6.4.1 There does not exist a uniform observation-respecting encoding from $\pi_m^{-\nu}$ into π_s.

Proof. By Proposition 6.2.16, Theorem 6.2.28 and Lemma 6.3.2.



π_s	=	Separate choice π-calculus
$\pi_m^{-\nu}$	=	Mixed choice π-calculus without restriction
MA^{io}	=	MA without communication primitives, restriction and open
MA^{-in}	=	MA without the in capability
MA^{ob}	=	MA with objective moves
SA^{iop}	=	SA without without communication primitives, restriction and out

Figure 6.1: Graphical view of separation results

6.4.2 *Comparing the π-calculus with ambients*

In this section the negative results aim to compare the relative strength of the the ambient calculi and the π-calculus dialects.

First we show that the ambient calculus without communication and open cannot be encoded into the π-calculus with separate choice.

COROLLARY 6.4.2 There does not exist a uniform observation-respecting encoding from MA^{io} into π_s.

Proof. By Proposition 6.2.17, Theorem 6.2.28 and Lemma 6.3.2.

On the other hand, the π-calculus with mixed choice admits a symmetric electoral system, which implies that there is no encoding from the π-calculus with mixed choice to MA without the in capability.

COROLLARY 6.4.3 There does not exist a uniform observation-respecting encoding from $\pi_m^{-\nu}$ into MA^{-in}.

Proof. By Proposition 6.2.16, Theorem 6.2.34 and Lemma 6.3.2.

Zimmer [92] has encoded [49] synchronous π-calculus without choice into SAwithout communication primitives. Below we show that the reverse is not possible.

COROLLARY 6.4.4 There does not exist a uniform observation-respecting encoding from SA^{iop} into π_s.

Proof. By Proposition 6.2.22, Theorem 6.2.28 and Lemma 6.3.2.

6.4.3 *Comparing ambient calculi*

This section aims to show that similar to the π-calculus world, also within the ambient world there is a hierarchy that is induced by the solution of the leader election problem in some dialects of the ambient calculus.

COROLLARY 6.4.5 There does not exist a uniform observation-respecting encoding from SA^{iop} into MA^{-in}.

Proof. By Proposition 6.2.22, Theorem 6.2.34 and Lemma 6.3.2.

COROLLARY 6.4.6 There does not exist a uniform observation-respecting encoding from MA^{io} into MA^{-in}.

Proof. By Proposition 6.2.17, Theorem 6.2.34 and Lemma 6.3.2.

6.4.4 *Objective moves*

One possible way of interpreting the result on MA with the in and out capabilities only, is that in the presence of tree the solution of a symmetrical electoral system is trivial. However it has been shown in Theorem 6.2.35 that this is not strictly true, since in presence of trees but without the ability to enter them there does not exist a symmetrical electoral system. One could regard the calculus without the in capability as an 'uninteresting' fragment of MA and still believe that symmetric electoral systems, in calculi with an inherent tree structure, are not interesting. We consider here a variant of the ambient calculus with *objective moves*. This variant of the calculus was first discussed by Cardelli and Gordon [24]. In this section we shall see that this dialect, which entails full tree structure similar to the standard MA, does not admit a solution for the leader election problem in symmetric networks.

Thus, we conjecture that the existence of electoral system has very little to do with the presence of trees, but more to do with the syntactic form of *redexes*. This last point needs clarification. In order to elect the leader in a symmetric network, the initial symmetry has to be broken. Intuitively, for this to happened, some possible computations have to be pre-empted. In reduction semantics, a term compute if it it contains an unguarded redex as subterm. Thus, one can look at breaking

symmetry from a syntactical point of view: certain form of redexes, such as the ones present in mixed choice π-calculus and MA, destroy joint redexes of contiguous processes. Consider a concrete example in MA:

$$P = n\,[\mathsf{in}\,q] \mid q\,[\mathsf{in}\,n]. \qquad (1)$$

The process P contains two redexes, and happens to be a symmetric network of size 2, with an automorphism $\sigma = \{n \mapsto q, q \mapsto n\}$. There are two features to observe: each subprocess of the network does not contain composition, and if one of the two redexes reduces, then the other is destroyed. Consider now the variant of the ambient calculus with objective moves, which we define MA^{ob}. The objective calculus has two different capabilities with respect to standard MA. Instead of $\mathsf{in}\,n$ and $\mathsf{out}\,n$ there are $\mathsf{mvin}\,n$ and $\mathsf{mvout}\,n$. We replace RED IN and RED OUT by the following reduction rules:

$$\mathsf{mvin}\,n.P \mid n\,[Q] \longrightarrow_o n\,[P \mid Q] \quad \text{RED OBJ-IN}$$
$$n\,[\mathsf{mvout}\,n.P \mid Q] \longrightarrow_o P \mid n\,[Q] \quad \text{RED OBJ-OUT}$$

Let's consider the form of redexes in this variant of the ambient calculus. We write what would be the equivalent network (1) written above.

$$\begin{aligned} Q_1 &= \mathsf{mvin}\,n.\mathbf{0} \mid q\,[\,]) \\ Q_2 &= \mathsf{mvin}\,q.\mathbf{0} \mid n\,[\,] \\ Q &= Q_1 \mid Q_2. \end{aligned}$$

The network Q contains two redexes as well, and similarly to P is symmetric to the automorphism σ written above; moreover there are two possible computations only. So far the two networks, P and Q enjoy the same properties. However, in Q each subprocess Q_1 and Q_2 contains composition and each redex can reduce without destroying the other. For this reason, we can think of the redex in the calculus with objective moves as symmetry-preserving. Similar observations can be made between the redexes of the π-calculus with mixed choice and with separate choice. We leave for future work the definition of a general format that would captures symmetry-preserving redexes.

In order for a symmetric network to be an electoral system there are two requirements:

i) Every computation has to break symmetry.

ii) There must be the possibility of displaying the winner.

Of course a system that never breaks symmetry cannot be an electoral system; however even when symmetry is broken, it may be the case that the winner cannot be announced. This would be the case for the ambient calculus with the in capability only. We have shown in Theorem 6.2.35 that MA without the in capability need never break the initial symmetry condition. Similarly to

the latter case, in the section below we shall prove that the objective MA need never break the initial symmetry

LEMMA 6.4.7 Let P be a process in MA^{ob} and σ a permutation. Then $P \downarrow n$ if and only if $\sigma(P) \downarrow \sigma(n)$.

Proof. The proof is identical to the one in Lemma 6.2.30.

LEMMA 6.4.8 Let P be a process in MA^{ob}, σ a substitution. and $Q{=}\sigma(P)$. Whenever $P \longrightarrow_o P'$ then for some Q', $Q \longrightarrow_o Q'$ and $Q'{=}\sigma(P')$.

Proof. By induction on $\longrightarrow_o$.

LEMMA 6.4.9 Let $\mathsf{N} = [P_0 \mid \ldots \mid P_{k-1}]$ be a symmetric network in MA^{ob}. Assume that $\mathsf{N} \longrightarrow_o \mathsf{N}^1$. Then there exists a computation $\mathcal{C} : \mathsf{N}^1 \longrightarrow\!\!\!\!\rightarrow_o \mathsf{N}^k$ such that:

i) N^k is symmetric;

ii) if for some i such that $0 \leq i \leq k-1$ we have $\mathsf{N}^1 \downarrow \omega_i$ and $\mathsf{N} \not\downarrow \omega_i$ then for all h such that $0 \leq h \leq k-1$ we have $\mathsf{N}^k \downarrow \omega_h$;

iii) if for all j such that $0 \leq j \leq k-1$ we have $\mathsf{N}^1 \not\downarrow \omega_j$ then for all t, j such that $1 \leq t \leq k-1$ and $0 \leq j \leq k-1$ we have $\mathsf{N}^t \not\downarrow \omega_j$.

Proof. Assume that N of size k is symmetric with respect to an automorphism σ with one orbit only and that $\mathsf{N} \longrightarrow_o \mathsf{N}^1$. There are two cases to consider:

Transition derived from one process only : In this case $\mathsf{N} \longrightarrow_o \mathsf{N}^1$ is the result of the computation of one process in the network. Assume P_r in N has reduced as follows $P_r \longrightarrow_o P_r^*$. The proof for this case is identical to Lemma 6.2.26.

Transition derived from the communication of two processes in the network :

In this case the computation has occurred between two different processes in the network. We assume, without loss of generality and to the mere end of simplifying the notation, that P_0 and P_d are the two processes involved in the computation. We will silently use structural congruence to move processes to adjacent positions in order to reduce and to move them back to their original positions. Moreover, in order to simplify the notation, we assume that

$$\sigma(0) = 1 \;\; \sigma^2(0) = 2 \ldots \sigma^d(0) = d \ldots \sigma^{k-1}(0) = k-1 \;\; \sigma^k(0) = 0$$

which implies by symmetry the following:

$$[\sigma(P_0) \mid \sigma^2(P_0) \mid \ldots \mid \sigma^{k-1}(P_0)]{=}[P_0 \mid P_{\sigma(0)} \ldots \mid P_{\sigma^{k-1}(0)}].$$

By the operational semantics of this calculus, there are three cases to consider.

1) : The reduction has occurred by using the rule RED OBJ-IN . Then without loss of generality we can assume that P_0 performs the entering into another ambient. Then the syntactic form of the two processes is:

$$\begin{aligned} P_0 &= (\nu p_1^0 \dots p_s^0)(\mathsf{mvin}\, n.S_0 \mid P_0') \\ P_d &= (\nu p_1^d \dots p_s^d)(n\,[Q_d] \mid P_d') \end{aligned}$$

with $n \notin \{p_1^0 \dots p_s^0\} \cup \{v_1^d \dots v_s^d\}$ (otherwise the two processes would not be able to compute).

i) By symmetry, we conclude the following statements starting from P_0.

$$\begin{aligned} P_{\sigma(0)} &= (\nu p_1^1 \dots p_s^1)(\mathsf{mvin}\, \sigma(n).\sigma(S_0) \mid \sigma(P_0')) \\ P_{\sigma^2(0)} &= (\nu p_1^2 \dots p_s^2)(\mathsf{mvin}\, \sigma^2(n).\sigma^2(S_0) \mid \sigma^2(P_0')) \\ &\vdots \\ P_{\sigma^{k-1}(0)} &= (\nu p_1^{k-1} \dots p_s^{k-1})(\mathsf{mvin}\, \sigma^{k-1}(n).\sigma^{k-1}(S_0) \mid \sigma^{k-1}(P_0')). \end{aligned}$$

Similar conclusions can be drawn starting from P_d.

$$\begin{aligned} P_{\sigma(d)} &= (\nu p_1^{d+1} \dots p_s^{d+1})(\sigma(n)\,[\sigma(Q_d)] \mid \sigma(P_d')) \\ P_{\sigma^2(d)} &= (\nu p_1^{d+2} \dots p_s^{d+2})(\sigma^2(n)\,[\sigma^2(Q_d)] \mid \sigma^2(P_d')) \\ &\vdots \\ P_{\sigma^{k-1}(d)} &= (\nu p_d^{d-1} \dots p_s^{d-1})(\sigma^{k-1}(n)\,[\sigma^{k-1}(Q_d)] \mid \sigma^{k-1}(P_d')). \end{aligned}$$

Hence, we conclude that each process in the network contains an in -redex. First, we analyse S_0 and P_d.

$$\begin{aligned} S_0 &= (\nu p_1^0 \dots p_s^0)(\mathsf{mvin}\, n.S_0 \mid \sigma^{-d}(n)\,[Q_0] \mid R_0) \\ P_d &= (\nu p_1^d \dots p_s^d)(\mathsf{mvin}\, \sigma^{d+d}(n).S_d \mid n\,[Q_d] \mid R_d). \end{aligned}$$

Now if $P_0 \mid P_d \longrightarrow_o P_0^+ \mid P_d^-$ then the residual processes must have the syntactical form:

$$\begin{aligned} P_0^+ &= (\nu p_1^0 \dots p_s^0)(\sigma^{-d}(n)\,[Q_0] \mid R_0) \\ P_d^- &= (\nu p_1^d \dots p_s^d)(\mathsf{mvin}\, \sigma^d(n).S_d \mid n\,[Q_d \mid S_0] \mid R_d). \end{aligned}$$

Therefore in general, for all h such that $0 \leq h \leq k-1$ we have

$$P_h = (\nu p_1^h \dots p_s^h)(\mathsf{mvin}\, \sigma^h(n).S_h \mid \sigma^{h-d}(n)\,[Q_h] \mid R_h).$$

and the residues of the processes are the following:

$$\begin{aligned} P_h^+ &= (\nu p_1^h \dots p_s^h)(\sigma^{h-d}(n)\,[Q_h] \mid R_h) \\ P_h^- &= (\nu p_1^h \dots p_s^h)(\mathsf{mvin}\, \sigma^d(n).S_h \mid \sigma^{h-d}(n)\,[S_{\sigma^{h-d}(0)} \mid Q_h] \mid R_h). \\ P_h^{+-} &= P_h^{-+} = (\nu p_1^h \dots p_s^h)(\sigma^{h-d}(n)\,[S_{\sigma^{h-d}(0)} \mid Q_h] \mid R_h). \end{aligned}$$

Now in order to show concretely how the computation goes around (similarly to what has been done in the previous proof Lemma 6.2.26), let $k - d = r$. Now we need to consider the relationship between r and d. There are two cases to consider: $r \leq d$ or $d < r$. We consider $d < r$ (the other case was considered in the proof of Lemma 6.2.26) in which case we let $m = r - d$ so $k = 2d + m$ (the other cases in which $hd + m = k$ for $0 \leq h \leq k$ are similar). Hence the computation proceeds as shown below:

$$\begin{array}{rcl}
P_1 \mid P_{d+1} & \longrightarrow_o & P_1^+ \mid P_{d+1}^- \\
P_2 \mid P_{d+2} & \longrightarrow_o & P_2^+ \mid P_{d+2}^- \\
 & \vdots & \\
P_{d-1} \mid P_{2d-1} & \longrightarrow_o & P_{d-1}^+ \mid P_{2d-1}^- \\
P_d^- \mid P_{2d} & \longrightarrow_o & P_d^{-+} \mid P_{2d}^- \\
P_{d+1}^- \mid P_{2d+1}^+ & \longrightarrow_o & P_{d+1}^{-+} \mid P_{2d+1}^- \\
 & \vdots & \\
P_{d+(m-1)}^- \mid P_{k-1} & \longrightarrow_o & P_{d+(m-1)}^{-+} \mid P_{k-1}^- \\
P_{2d}^{-+} \mid P_0^+ & \longrightarrow_o & P_{d+m}^{-+} \mid P_0^{+-} \\
 & \vdots & \\
P_{k-1}^- \mid P_{d-1}^+ & \longrightarrow_o & P_{k-1}^{-+} \mid P_{d-1}^{+-}.
\end{array}$$

Then the network is partitioned in this way:

$$\mathsf{N}^h = [P_0 \mid \ldots \mid P_d \mid \ldots \mid P_{2d} \mid \ldots \mid P_{k-1}].$$

The computation of the network after the initial step is the following:

$$\begin{array}{lcl}
\mathsf{N}^1: & [P_0^+ \mid P_1 \mid \ldots \mid P_d^- \mid P_{d+1} \ldots \mid P_{k-1}] & \longrightarrow_o \\
\mathsf{N}^2: & [P_0^+ \mid P_1^+ \mid \ldots \mid P_d^- \mid P_{d+1}^- \mid \ldots \mid P_{k-1}] & \longrightarrow_o \\
 & & \vdots \\
\mathsf{N}^d: & [P_0^+ \mid P_1^+ \mid \ldots \mid P_d^{-+} \mid \ldots \mid P_{2d}^- \mid \ldots \mid P_{k-1}] & \longrightarrow_o \\
 & & \vdots \\
\mathsf{N}^k: & [P_0^{+-} \mid P_1^{+-} \mid \ldots \mid P_d^{-+} \mid \ldots \mid P_{2d}^{-+} \mid \ldots \mid P_{k-1}^{-+}]. &
\end{array}$$

Hence, we have $\mathsf{N}^1 \longrightarrow\!\!\!\!\rightarrow_o \mathsf{N}^k$.

Now we have to show that the network N^k is symmetric.

Now, N^k is symmetric with respect to σ, since for all s such that $0 \leq s \leq k - 1$ the

following holds:

$$\begin{aligned} P^{+-}_{\sigma^h(0)} &= (\nu p^h_1 \ldots p^h_s)(\sigma^{h-d}(n)\ [S_{h-d} \mid Q_h] \mid R_h) \\ &= (\nu p^h_1 \ldots p^h_s)(\sigma^{h-d}(n)\ [S_{\sigma^{h-d}(0)} \mid Q_{\sigma^h(0)}] \mid R_{\sigma^h(0)}) \\ &= (\nu p^h_1 \ldots p^h_s)(\sigma^{h-d}(n)\ [\sigma^h(S_{\sigma^{-d}(0)}) \mid \sigma^h(Q_0)] \mid \sigma^h(R_0)) \\ &= \sigma^h((\nu p^0_1 \ldots p^0_s)(\sigma^{-d}(n)\ [S_{\sigma^{-d}(0)} \mid Q_0] \mid R_0)) \\ &= \sigma^h(P^{+-}_0) \end{aligned}$$

σ has one orbit only since it has not changed.

ii) The proof for this part of the lemma is similar to that of Lemma 6.2.33.

iii) Observe that for this particular reduction rule the following holds: if for all t there exists a h such that $0 \leq t, h \leq k-1$ we have $\mathsf{N}^{t+1} \downarrow \omega_h$ then $\mathsf{N} \downarrow \omega_h$. Thus, by assumption for all i $0 \leq i \leq k-1$ we have $\mathsf{N}^1 \not\downarrow \omega_i$ we conclude that for all t such that $0 \leq t \leq k-1$ the following holds $\mathsf{N}^t \not\downarrow \omega_i$.

2) : In the case in which the reduction is derived from an open -redex by means of the rule RED OPEN, then the proof is identical to Lemma 6.2.33.

3) : If the reduction is triggered by the rule RED COMM, this means the redex is formed with communication primitives, and then the case is identical to the proof supplied in Lemma 6.2.33 for the rule RED COMM.

LEMMA 6.4.10 Let $\mathsf{N} = [P_0 \mid \ldots \mid P_{k-1}]$ with $k \geq 2$ be a symmetric network in MA^{ob}. Then there exists a computation $\mathcal{C}$ such that either $\mathsf{Obs}(\mathcal{C}) = \{\omega_0, \omega_1, \ldots, \omega_j, \ldots, \omega_{k-1}\}$ or $\mathsf{Obs}(\mathcal{C}) = \emptyset$.

Proof. The proof is identical to that of Lemma 6.2.27.

THEOREM 6.4.11 Let $\mathsf{N} = [P_0 \mid \ldots \mid P_{k-1}]$ be a symmetric network in MA^{ob} with $k \geq 2$. Then N cannot be an electoral system.

Proof. The proof is identical to that of Theorem 6.2.28 by using Lemma 6.4.10.

COROLLARY 6.4.12 There does not exist a uniform observation-respecting encoding from MA^{io} into MA^{ob}, or from π_m into MA^{ob}.

6.5 *Concluding remarks*

The work carried out in this chapter has a wide applicability to similar process calculi, such as Boxed Ambients, Seal Calculus, etc. From Theorem 6.2.19 it should follow trivially that Boxed Ambients [14] can solve the leader election problem in symmetric networks. We conjecture that the Seal calculus [88] should also admit a solution the the leader election problem, since the (move in) seems to be a *symmetry-breaking* operator. On the other hand we speculate the pure version of $D\pi$ [42] cannot solve such a problem.

In [34] there exists an encoding of MA into the Distributed Join Calculus which has led to the implementation of MA into Jocaml. For that encoding, the migration of ambients, entering and exiting, have be separated into three atomic steps. Moreover, each migration of ambients happens via the centralized control of the parent. For example, if $a\,[\mathsf{in}\,b]$ wants to migrate to the sibling $b\,[\,]$, then (1) has to forward a request to the parent ambient $c\,[\,]$; (2) the ambient c informs b of the request, and then the final action is that the ambient b becomes the parent of a. The ambient c deals with all the requests regarding the child-ambients, thus we cannot regard this encoding as fully distributed in the sense specified by Definition 6.3.1, since the the encoding of the electoral system in Proposition 6.2.17 would make use of a parent ambient to deal with the request of entering. This can be regarded as introducing a centralized server, which makes the solution of electing a leader trivial.

We regard the work done in this Chapter as the starting point in order to explore further the relationship between the π-calculus and new generation of process calculi.

Conclusions

Summary

The overall purpose of this thesis has been to deepen the understanding of Mobile Ambient. This is a process calculus that aims to represent proper features of computation over the World Wide Web such as locations, code mobility, devices mobility, abstract domains and some form of security. Overall we have taken into account three different dialects that seem significant: Mobile Ambient [24], Safe Ambients [49] and the Push and Pull Ambient Calculus [73]. Mobile Ambients, describes migration of agents quite naturally; the main drawback is that processes are not in control of their own boundaries. Safe Ambients gives control on the boundaries of process, but it assumes a synchronisation over agents which unrealistic. The Push and Pull Ambient Calculus achieves control on boundaries of domains without the synchronisation at the prize to represent a more traditional client-server architecture. In this dissertation, We have carried out our research in the reduction semantics frameworks, which we believe to be a suitable semantics. This framework has been generally defined and motivated in Chapter 1.

In Chapter 2 we have revised the literature. The π-calculus and of its some its variant (the π-calculus with mixed choice, the π-calculus with separate choice, asynchronous π-calculus) had been introduced, together with Mobile Ambients MA and Safe Ambients SA.

In Chapter 3 we argued that MA is very expressive since the matching operator is faithfully encodable, without any loss of expressiveness. The encoding is strong since homomorphic and fully abstract. We discuss the non-encodability of the matching operator in the π-calculus with mixed choice and in the asynchronous π-calculus. We then saw that pure MA cannot be encoded in the π-calculus without matching. An extended example of a router delivering e-mails was given.

In Chapter 4 we proposed a variant of MA, called Push and Pull Ambient Calculus PAC, in order to represent Internet scenarios where the control of the boundaries lies in the ambient. Some application of this can be a resource sensitive example such as banking. We have shown that this calculus is as expressive as standard MA in the sense that many of the examples presented in [24] can be encoded in PAC as well. We have also defined a well-behaved encoding of the asynchronous π-calculus and of the matching operator.

The aim of Chapter 5 was to deepen the understanding of reduction semantics for MA by applying the Honda and Yoshida technique. Contextual barbed bisimulation can be derived by

a small set of equations in MA, as was done for the π-calculus in [46]. We have applied his techniques to PAC and SA as well. In particular for SA, we have seen that contextual barbed congruence equate the same terms regardless of the barbs observed. This is not true for MA or PAC.

Expressiveness results, i.e. comparing the strength of calculi, are usually hard to achieve. Particularly hard is proving that encoding do not exist. This is the task that we set for the last chapter of this dissertation. We have investigated the relative strengths of the π-calculus and the Mobile Ambients. We have used electoral systems to show that a fragment of Mobile Ambients without communication primitives, and open cannot be encoded in the π-calculus with separate choice (under certain natural conditions on encodings). We have seen that this is due to the symmetry-breaking power of the in capability of MA (a power shared with the mixed choice operator of the π-calculus). In fact the calculus MA without the in capability, which is basically Mobile Ambients without the in cannot solve leader election problems. By analysis which fragment can solve the leader election problem, we have been able to give a hierarchy within the family of ambient calculi. We end this dissertation by sketching how we envisage future work.

Future work

Research is a journey that might never end. Answers to questions produce new questions, and therefore more research. The work presented here is part what has been produced within the time constraints. However, more could be done in future work. In this final section we shall discuss future work for each chapter of this dissertation.

Chapter 1 : In this chapter we have introduced in a general way a body of definitions for reduction semantics. Although the definitions are clear and general enough, we have assume the existence of a general grammar. This means that directly or indirectly, we have appealed to the intuition of the reader. We believe that in order to finish the work, it would be nice to define a general grammar, or a general format, similar to the De Simone format, that encompasses both the π-calculus and the various dialects of the ambient calculus. We believe that at the heart of those calculi lies a clear and neat general definition of binding operators, like for example restriction ν and input $m(y).P$). We envisage that the work on the logic for binding operators, by Gabbay and Pitts [35], will be useful in this direction.

Chapter 2 and Chapter 4 : In these chapters we have presented different ambient calculi. In the grammar, the ambient calculus uses replication $!P$ instead of recursion $rec(X).P$. It is a well-known result that in (any dialect of) the π-calculus the two operators are equivalent

[78], however, the matter is not solved for the general case of MA or any of its dialects. Busi and Zavattaro [16] proved that for a fragment with the open only, recursion is strictly more expressive than replication. This is not surprising, since this fragment is basically CCS without choice. It is an open and challenging problem to show that in the full MA (or SAor PAC) recursion is more expressive than replication.

Chapter 3 : In this chapter we have defined the extended example of the router that deliver e-mails. The router read the address and translated it into the real location. Since the ambient calculus has very good primitives for representing administrative domains, it would be interesting to consider real-life Inter-Administrative Protocols and model those in one of the dialects of the ambient calculus. The final goal of such an operation would be to verify correctness of the protocol. We envisage a solution by using some typing system such as the one devised in [27], to the end of representing the information about policies.

Chapter 5 : We have deepened our understanding of the reduction semantics. Contextual barbed bisimulation come equipped with coinductive proof techniques, which means that in order to prove that two process are contextual bisimilar, one has to exhibit a contextual barbed bisimulation. Often to show that a relation is a contextual bisimulation can be quite hard. In [90], Yoshida proposed some proof techniques to relieve the complexity of such proofs. Future work goes in the direction of adapting these proof techniques to the Ambient Calculus and Safe Ambients.

More challenging would be to show that the barbed congruence is included into the contextual barbed bisimulation. In [33] there is the only direct proof done by using reduction semantics for the asynchronous π-calculus. The techniques used there are quite involving and the definition of an interpreter, or universal machine, is needed. For the asynchronous π-calculus the definition of interpreter involved the definition of numeral, tests on numerals. In the π-calculus these definitions are quite natural [33, 78]. We believe that for the PAC (and SA) a possible solution could make use of the encoding of the asynchronous π-calculus, and inherit some datatype from there such as numerals, lists, etc. A preliminary definition for PAC can be found in [84]. It remains a challenge to define a self-interpreter in the standard ambient calculus, where the encoding of the π-calculus is not helpful.

Since we have adapted the Honda-Yoshida method based on active names to ambient calculi, an open question is whether it is possible to generalize this definition to a general grammar, that encompasses both the ambient calculi and the π-calculus. Results such as Theorem 5.4.22 and Theorem 5.4.23 should then directly provable in this framework, and derivable for concrete languages.

Although the Honda-Yoshida framework is very interesting and elegant, the main disadvantage of working with active names is the lack of compositional reasoning. It would be interesting to devise some type system that captures the notion of insensitive terms.

Chapter 6 : We have seen that certain calculi can solve leader election problems in the presence of symmetric networks and others can not. One way to approach this in a broader spectrum, is to categorize operators in languages, such as *symmetry-breaking* and *symmetry-preserving.* Clearly, by Theorem 6.2.35, in MA entering another ambient is symmetry breaking; however it is *symmetry-preserving* in objective MA by Theorem 6.4.11. Similarly for the π-calculus, mixed choice is *symmetry-breaking* while separate choice is *symmetry-preserving.* It is a challenge to investigate this problem from an abstract approach (i.e. independently from any specific language). This would involve defining some sort of format that encompasses *symmetry-preserving* operators.

In chapter 6§6.2.1 we proved that MA with the in and out only solves the election leader problem. This calculus does not contain open, restriction, and communication capabilities. We conjecture that this is the minimal calculus that can solve the leader election problems. This result needs proof.

For our predecessors [12, 70, 28, 71], results on symmetric networks were formulated in a graph framework. In other words, one could define a mapping from network to graphs, and analyse leader election problems in different graphs. This was studied by [3]. In this dissertation, we have shown that graphs are not necessary for defining leader election problems in symmetric networks, and for proving our separation results. If we were using graphs, in this thesis, it would be fair to say that we have been mostly working on fully connected graphs. However, it remains to study electoral systems on graphs that are not fully connected for MA. It should be possible to solve the problem in a ring, in the presence of communication primitives and open, since one could simulate channelled communication. Thus an election leader in a ring could be reduced to the problem of electing a leader in fully connected graph, as Palamidessi's algorithms shows. We conclude that one challenge seems related to solve the election leader problem in rings in without the help of communicating primitives.

References

[1] M. Abadi and A.D. Gordon. A calculus for cryptographic protocols: the spi-calculus. *Information and Computation*, 148:1–70, 1999.

[2] T. Amtoft, A.J. Kfoury, and S.M. Pericas-Geertsen. What are Polymorphically-Typed Ambients? In *Proceedings of ESOP'01*, volume 2028, pages 206–220, 2001.

[3] D. Angluin. Local and global properties in networks of processors. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing*, pages 82–93. ACM, 1980.

[4] H. Attiya and J. Welch. *Distributed Computing*. The McGraw-Hill Companies, 1998.

[5] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge University Press, 1990.

[6] H.P. Barendregt. *The lambda calculus.* North-Holland, 1991.

[7] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Computation*, 60(1/3):109–137, 1984.

[8] G. Berry and G. Boudol. Chemical Abstract Machine. In *Proceedings of the 17th ACM Symposium on Principles of Programming Languages*, pages 81–94. ACM, 1990.

[9] M. Boreale, R. De Nicola, and R. Pugliese. Trace and Testing Equivalence on Asynchronous Processes. *Information and Computation*, 172:139–164, 2002.

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

[11] G. Boudol, I. Castellani, M.C.B. Hennessy, and A. Kiehn. A Theory of Processes with Localities. *Formal Aspects of Computing*, 6:171–200, 1994.

[12] L. Bougé. On the existence of symmetric algorithms to find the leaders in networks of communicating sequential systems. *Acta Informatica*, 25:179–201, 1988.

[13] C. Braghin, A. Cortesi, and R. Focardi. Control Flow Analysis of Mobile Ambients with Security Boundaries. In *Proceedings of FMOODS'02*, pages 197–212. Kluwer Academic Publisher, 2002.

[14] M. Bugliesi, G. Castagna, and S. Crafa. Boxed ambients. In *Proceedings of TACAS'01*, volume 2215 of *LNCS*, pages 38–63. Springer-Verlag, 2001.

[15] N. Busi and G. Zavattaro. On the Expressiveness of Movement in Pure Mobile Ambients. In *Proceedings of F-WAN'02*, volume 66 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.

[16] N. Busi and G. Zavattaro. On the Expressive Power of Movement and Restriction in Pure Mobile Ambients. *Theoretical Computer Science*, 2004. To appear.

[17] M. Carbone and S. Maffeis. On the Expressive Power of Polyadic Synchronisation in π-calculus. *Nordic Journal of Computing*, 10(2):70–98, 2003.

[18] L. Cardelli. Global Computing. *ACM Computing Surveys*, 28A(4), 1996.

[19] L. Cardelli. Abstraction for Mobile Computation. In J. Vitek and C. Jensen, editors, *Proceedings of Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, volume 1603 of *LNCS*, pages 51–94. Springer-Verlag, 1999.

[20] L. Cardelli, G. Ghelli, and A.D. Gordon. Mobility Types for Mobile Ambients. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, volume 1644 of *LNCS*, pages 230–239, April 1999.

[21] L. Cardelli and A.D. Gordon. Mobile ambients. In *Proceedings of FoSSaCS'98*, volume 1378 of *LNCS*, pages 140–155. Springer-Verlag, 1998.

[22] L. Cardelli and A.D. Gordon. Types for Mobile Ambients. In *Proceedings of the 26th ACM Symposium on Principles of Programming Languages*, pages 79–92. ACM, January 1999.

[23] L. Cardelli and A.D. Gordon. Anytime, Anywhere. Modal Logic for Mobile Ambients. In *Proceedings of the 27th ACM Symposium on Principles of Programming Languages*, pages 365–377, 2000.

[24] L. Cardelli and A.D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000.

[25] W. Charatonik, A.D. Gordon, and J. Talbot. Finite control ambients. In *Proceedings of ESOP'02*, volume 2305 of *LNCS*, pages 295–313. Springer-Verlag, 2002.

[26] T. Chothia and I. Stark. Encoding distributed areas and local communication into the pi-calculus. In *Proceedings of HLCL'00*, volume 52 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2000.

[27] M. Dezani-Ciancaglini and I. Salvo. Security types for safe mobile ambients. In *Proceedings of ASIAN'00*, pages 215–236. Springer-Verlag, 2000.

[28] C. Ene and T. Muntian. Expressiveness of point-to-point versus broadcast communication. In *Proceedings of FCT'99*, volume 1684 of *LNCS*, pages 258–268. Springer-Verlag, 1999.

[29] D. Estrin. Policy Requirements for Interadministrative Domain Routing. Network Working Group, Internal report University Southern California, 1989.

[30] G. Ferrari, U. Montanari, and E. Tuosto. A LTS Semantics of Ambients via Graph Synchronization with Mobility. In *Proceedings of ICTCS'01*, volume 2202 of *LNCS*, pages 1–16. Springer, 2001.

[31] C. Fournet. *The Join-Calculus: A Calculus for Distributed Mobile Processes.* PhD thesis, École Polytechnique, Paris, 1998.

[32] C. Fournet and G. Gonthier. A Hierarchy of Equivalences for Asynchronous Calculi. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, volume 1443 of *LNCS*. Springer-Verlag, 1998.

[33] C. Fournet and G. Gonthier. A Hierarchy of Equivalences for Asynchronous Calculi. *JLAP: special issue on the π-calculus*, 2004. To appear.

[34] C. Fournet, J.J., Lévy, and A. Schmit. An Asynchronous, Distributed implementation of Mobile Ambient. In *Proceedings of IFIP International Conference on Theoretical Computer Science*, volume 1872, pages 348–364, August 2002.

[35] M.J. Gabbay and A.M. Pitts. A New Approach to Abstract Syntax Involving Binders. In *14th Annual Symposium on Logic in Computer Science*, pages 214–224. IEEE Computer Society Press, Washington, 1999.

[36] A.D. Gordon. Bisimilarity as a Theory of Functional Programming. *Theoretical Computer Science*, 228:5–47, 1999.

[37] A.D. Gordon. Notes on nominal calculi for security and mobility. In *Proceedings of Foundations of Security Analysis and Design*, volume 2171 of *LNCS*, pages 262–330. Springer-Verlag, 2002.

[38] A.D. Gordon and L. Cardelli. A Commitment Relation for the Ambient Calculus. Available at: www.microsoft.com/~adg, 1998.

[39] A.D. Gordon and L. Cardelli. Equational Properties of Mobile Ambients. *Mathematical Structures in Computer Science*, 13:371–408, 2003. Also available as Technical Report, MSR-TR-99-11, Microsoft Research Center.

[40] L. Guan, Y. Yang, and J. You. Making ambients more robust. In *Proceedings of International Conference on Software: Theory and Practice*, pages 377–384, 2000. Beijing, China.

[41] C.A. Gunter. *Semantics of Programming Languages: Structures and Techniques.* Foundations of Computing. MIT Press, 1992.

[42] M. Hennessy and J. Riely. Resource access control in systems of mobile agents. In *Proceedings of HLCL'98*, volume 16.3 of *Electronic Notes in Theoretical Computer Science*, pages 3–17. Elsevier Science Publishers, 1998.

[43] D. Hirschkoff, É. Lozes, and D. Sangiorgi. Separability, Expressiveness and Decidability in the Ambient Logic. In *Proceedings of 17th Annual IEEE Symposium on Logic in Computer Science*, pages 423–431. IEEE Computer Society, 2002.

[44] C.A.R. Hoare. *Communicating sequential processes.* Prentice Hall, 1985.

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

[46] K. Honda and N. Yoshida. On the Reduction-based Process Semantics. *Theoretical Computer Science*, 151:437–486, 1995.

[47] A. Ingólfsdóttir and H. Lin. *A Symbolic Approach to Value-Passing Processes*, chapter 7, pages 27–478. Handbook of Process Algebra. North-Holland, 2001.

[48] F. Levi and D. Sangiorgi. Controlling Interference for Ambients. In *Proceedings of 28th ACM Symposium on Principles of Programming Languages*, 2000.

[49] F. Levi and D. Sangiorgi. Controlling IMerronterference for Mobile Safe Ambients. *ACM Transactions on Programming Languages and Systems*, 25(1):1–69, 2003.

[50] N.A. Lynch. *Distributed Algorithms.* Morgan Kaufmann, 1996.

[51] S. Maffeis and I.C.C. Phillips. On the Computational Strength of Pure Ambient Calculi. In *Proceedings of Express'03*, volume 91 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2004.

[52] M. Merro and M. Hennessy. Bisimulation Congruences in Safe Ambients. In *Proceedings of the 29th ACM Symposium on Principles of Programming Languages*, pages 71–80. ACM, 2002.

[53] M. Merro and F. Zappa Nardelli. Bisimulation Proof Methods for Mobile Ambients. In *Proceedings of ICALP'03*, volume 2719 of *LNCS*, pages 584–598. Springer, 2003.

[54] R. Milner. *Communication and Concurrency*. Prentice-Hall International, 1989.

[55] R. Milner. The polyadic π-calculus: a tutorial. Technical Report ECS-LFCS-919180, LFCS, Dep. of Computer Science, Edinburgh Uni., October 1991.

[56] R. Milner. Function as processes. *Mathematical Structures in Computer Science*, 2(2):269–310, 1992.

[57] R. Milner. The polyadic π-calculus: a tutorial. In *Proceedings of International NATO Summer School*, pages 428–440, 1994.

[58] R. Milner. *Communicating and Mobile Systems: the π-calculus.* Cambridge University Press, 1999.

[59] R. Milner. Bigraphical reactive systems. In *Proceedings of 12th International Conference in Concurrency Theory*, volume 2154 of *LNCS*, pages 16–35, 2001.

[60] R. Milner. Bigraphical reactive systems: basic theory. Technical Report 503, Computer Laboratory Cambridge University, 2001.

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

[62] R. Milner and D. Sangiorgi. Barbed Bisimulation. In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, volume 623 of *LNCS*, 1992.

[63] U. Montanari and V. Sassone. Dynamic congruence vs. progressing bisimulation for CCS. *Fundamenta Informaticae*, 16(2):171–199, 1992.

[64] J.H. Morris. *Lambda-Calculus Models of Programming Languages*. PhD thesis, M.I.T., 1968.

[65] U. Nestmann. What is a 'Good' Encoding of Guarded Choice? *Information and Computation*, 156:287–319, 2000.

[66] U. Nestmann. Modeling consensus in a process calculus. In *Proceedings of CONCUR'03*, volume 2761 of *LNCS*. Springer-verlag, 2003.

[67] U. Nestmann and B.C. Pierce. Decoding Choice Encoding. *Information and Computation*, 163(1):1–59, 2000.

[68] R. De Nicola and M.C.B. Hennessy. Testing Equivalence for Processes. *Theoretical Computer Science*, 34:83–133, 1984.

[69] H.R. Nielsen and F. Nielsen. Validating the firewalls in mobile ambients. In *Proceedings of CONCUR'99*, LNCS, pages 463–477, 1999.

[70] C. Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous π-calculus. In *Proceedings of the 25th ACM Symposium on Principles of Programming Languages*, pages 256–265. ACM, 1997.

[71] C. Palamidessi. Comparing the Expressive Power of the the Synchronous and Asynchronous π-calculus. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003.

[72] J. Parrow. *Introduction to the π-calculus*, chapter 8, pages 479–543. Handbook of Process Algebra. North-Holland, 2001.

[73] I.C.C. Phillips and M.G. Vigliotti. On the Reduction Semantics for the Push and Pull ambient calculus. In *Proceedings of IFIP International Conference on Theoretical Computer Science*, volume 1872, pages 550–562, August 2002.

[74] I.C.C. Phillips and M.G. Vigliotti. Electoral Systems in Ambient Calculi. In *Proceedings of FoSSaCS'04*, volume 2987 of *LNCS*, pages 408–422. Springer-verlag, 2004.

[75] K.V.S. Prasad. Broadcast Calculus Interpreted in CCS up to Bisimulation. In *Proceedings of Express'01*, Electronic Notes in Theoretical Computer Science. Elsevier, 2001.

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

[77] D. Sangiorgi. Extensionality and intensionality of the ambient logics. In *Proceedings POPL'01*, pages 4–13. ACM, 2001.

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

[79] P. Sewell. From Rewrite Rules to Bisimulation Congruences. Technical Report 498, Computer Laboratory Cambridge University, June 1998.

[80] G. Tel. *Distributed Algorithms*. Cambridge University Press, 2000.

[81] B. Thomsen. Calculi for Higher Order Communicating Systems. In *Proceedings of the 18th ACM Symposium on Principles of Programming Languages*, pages 143–154. ACM Press, January 1989.

[82] B. Thomsen. *Calculi for Higher Order Communicating Systems*. PhD thesis, Imperial College, 1990.

[83] R.J. van Glabbeek. *The Linear Time - Branching Time Spectrum I*, chapter 1, pages 3–99. Handbook of Process Algebra. North-Holland, 2001.

[84] M.G. Vigliotti. Universal Machine for Push and Pull Ambient Calculus. Available at: www.doc.ic.ac.uk/~mgv98.

[85] M.G. Vigliotti. Transition Systems for the Ambient Calculus. Master's thesis, Imperial College, 1998-99.

[86] M.G. Vigliotti. Semantics and expressiveness of the Ambient Calculus. Technical report, Imperial College, 2001. Ph.D. Transfer Report.

[87] M.G. Vigliotti and I.C.C. Phillips. Barbs and Congruences for Safe Mobile Ambients. In *Proceedings of F-WAN'02*, volume 66 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.

[88] J. Vitek and G. Castagna. Seal: A framework for secure mobile computation. In *Proceedings of Internet Programming Languages*, volume 1686 of *LNCS*, pages 47–77. Springer-Verlag, 1999.

[89] P.T. Wojciechowski. *Nomadic Pict: Language, Infrastructure Design for Mobile Computation*. PhD thesis, University of Cambridge, Cambridge, 2000.

[90] N. Yoshida. Graph Types for Monadic Mobile Process Calculi. Technical report, LFCS, 1996. ECS-LFCS-96-3502.

[91] N. Yoshida. Graph Types for Monadic Mobile Process Calculi. In *Proceeding of 16th FST/TCS*, volume 1180 of *LNCS*, pages 371–386. Springer-Verlag, 1996.

[92] P. Zimmer. On the Expressiveness of Pure Safe Ambients. *Mathematical Structures in Computer Science*, 13(5):721–770, 2003.